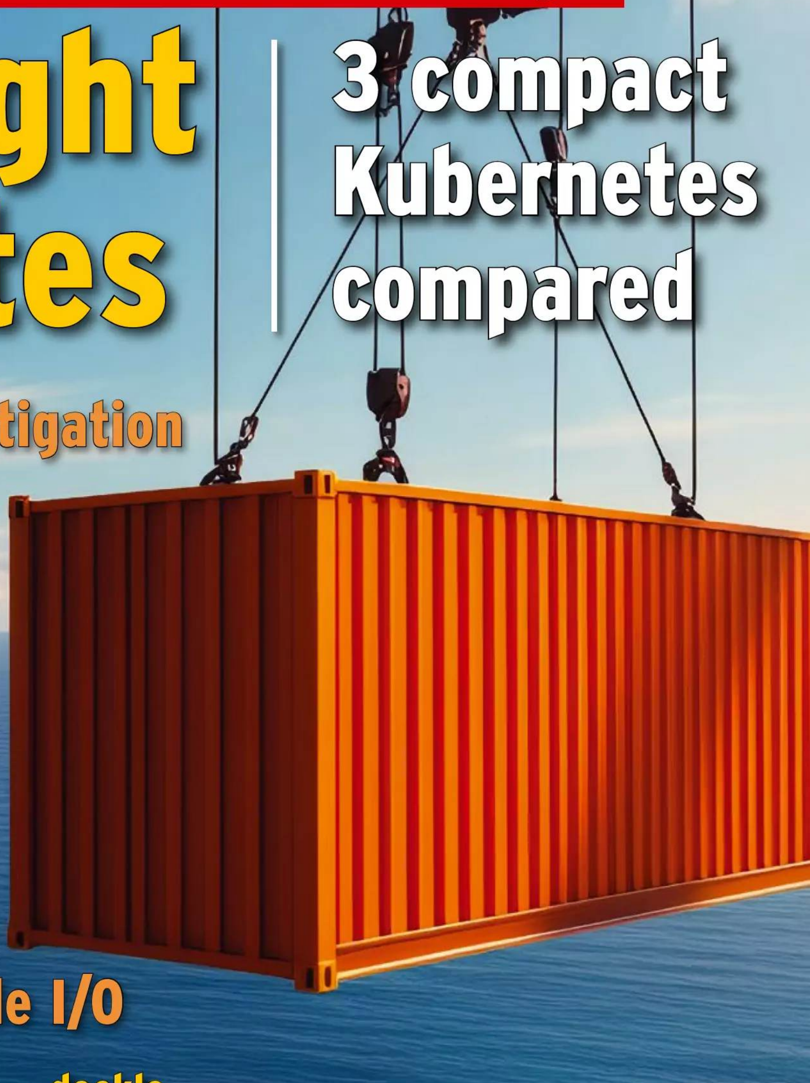
ADMIN Network & Security

Lightweight Kubernetes | **AD Risk Mitigation** | **IVRE**

ADMIN

Network & Security

ISSUE 87



Lightweight Kubernetes

3 compact Kubernetes compared

Active Directory Risk Mitigation

IVRE

Visualize IT infrastructures

Software Licenses

Coroot

Monitoring and attack detection with eBPF

Analyze and Optimize File I/O

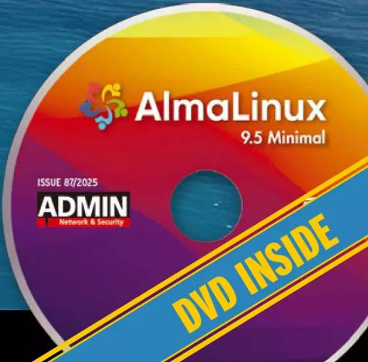
NAPS2 SDK
Develop a simple app for scanning documents

dockle
Keep your Docker containers compliant

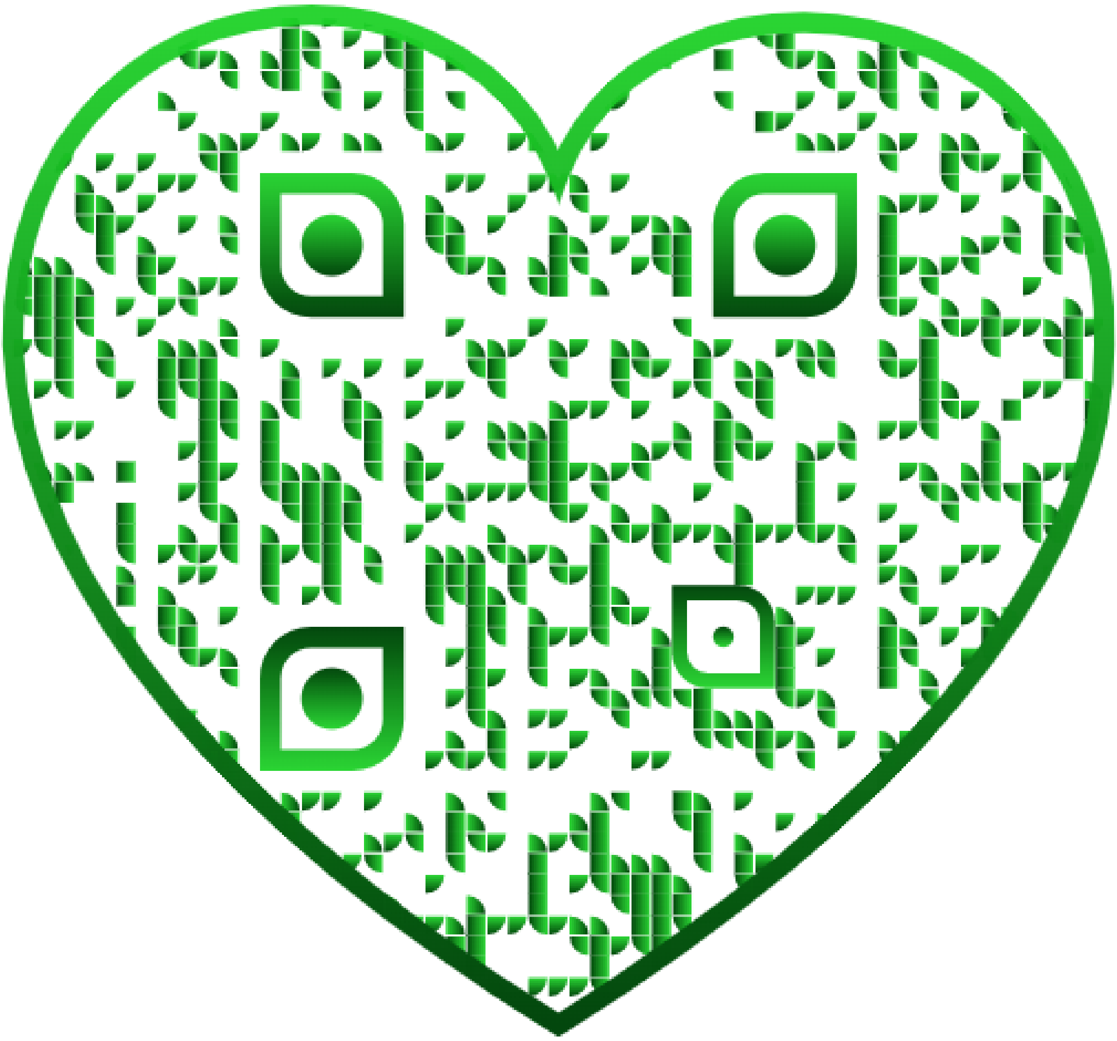
Netdata
Monitor your infrastructure

WireGuard Portal
Set up, manage, and scale a WireGuard VPN environment

LINUX NEW MEDIA
The Pulse of Open Source


AlmaLinux
9.5 Minimal
ISSUE 87/2025
ADMIN
Network & Security
DVD INSIDE

WWW.ADMIN-MAGAZINE.COM



LINUX-OPTIMIZED HARDWARE
FROM LAPTOP TO HIGH-END PC

TUXEDO develops and manufactures hardware specifically optimized for Linux – from powerful laptops and versatile desktops to workstations and mini PCs. Our wide range offers customized solutions for beginners, professionals, and businesses alike. Whether you're looking for a rugged a sleek ultrabook, or a high-end PC for demanding workloads – we have the perfect device, precisely tailored to your unique needs and preferences. 100% Linux-compatible, individually configurable, with an optional personalized cover logo and comprehensive, dedicated support!

Linux compatible
Up to 5 Years Guarantee
Immediately ready for use
TUXEDO
tuxe.do/lxadmin87
Made in Germany
German Data Privacy
German Tech Support

Self-Worth

Exploring my worth as a system administrator.

I love solving problems. I really do. Fixing something that's broken can be monotonous and boring, but it can also be challenging and rewarding. Solving a problem with a piece of community-built, open source software is a special thrill. No, seriously, it is. I love seeing happy faces when I present business managers, their users, and customers with a free solution to an often very expensive problem. I know that it's extremely nerdy of me to be thrilled to do this sort of thing, but it gives me a feeling of accomplishment. It makes me glad to do what I do.

You should explore Erik Erikson's Stages of Psychosocial Development, introduced in his book *Childhood and Society* (W W Norton & Co, 1st ed. 1950; 2nd ed. 1993), to understand this feeling. For working-aged people, say from 21 to 65 or 70, it explains a lot of what you'll experience in your life. In a nutshell, here are the three stages that affect most of us:

- 21-39 years: Intimacy vs. Isolation
- 40-65 years: Generativity vs. Stagnation
- 65 years and older: Ego Integrity vs. Despair

I'm in the 40-65 age range, where there is a need to connect and to contribute through parenting, mentoring, and community involvement. Success during this time leads to feelings of a sense of purpose and fulfillment. However, if you don't find success during these years, you might feel that you have no purpose or that you're unproductive. You might also think of it in system administration terminology of job satisfaction vs. burnout.

I've written about this topic before, but I think you need to evaluate or re-evaluate your life's purpose, your job, and your legacy; you don't have to be rich, famous, or a celebrity to have a legacy.

Being proud of what you've done is a good thing. I haven't always felt so good about my career or existence, so don't think that I'm coming at you from some lofty place. I've slipped into stagnation, burnout, and despair. For many years, I believed that I was simply in survival mode rather than thriving, believing in myself, or having a sense of worth. I think working with computers does that to us. I think that lack of human contact can isolate us. I think that doing crazy things with electronic blips on a screen gives us a sense of purposelessness in some cases. I'm not sure if purposelessness is a real word, but it conveys the meaning appropriately. I think you understand.

Don't confuse generativity with "being happy." No one is happy, excited, joyful, sad, angry, disappointed, or unsatisfied all the time. Life is incremental. We experience life in moments. We build on experiences, but we live life in bits and bytes, and in keeping with the computer theme: garbage in, garbage out. In other words, if you want to enjoy your life and your life's work, you must put your energy into doing your best.

You must decide that your job is your job. It is not you. You are not it. Doing your job well will make you feel good about yourself. Success is measured in how well you do it, not necessarily in how much money you make. Although money can certainly make you somewhat less stressed about some things, it doesn't bring happiness or contentment.

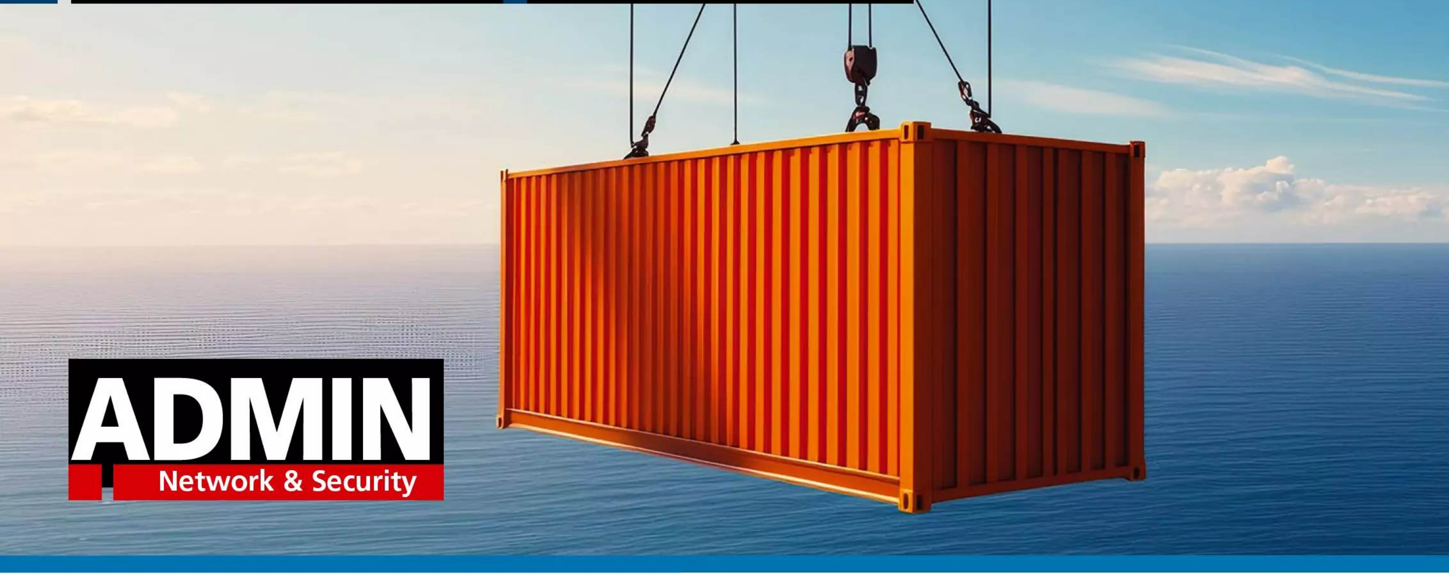
Remember that your job is ultimately focused on helping other people do something that they need or want to do. Whether you help programmers create new software, keep servers alive so that customers can buy groceries, or support gaming systems, you have a purpose: to help others. That is the part that makes me happy. Some people create art. Some operate on hearts. Some cook food. Some of us fix computers, install software, and keep businesses running. Our contributions are no less important than any others.

I only had to look at Erikson's final stage to know where I want to be in the Ego Integrity vs. Despair age. I want to look back on my legacy of writing words, rebooting computers, creating user accounts, and performing backups as a contribution to a world that needed what I had to offer. This is my legacy.

I fix things, I help people, I solve problems, I am a system administrator, and I am happy.

Ken Hess • Senior ADMIN Editor

Lead Image © xtrekx, 123RF.com



Features

10 **Lightweight Kubernetes**
We compare the k0s, K3s, and MicroK8s distributions for environments operating under resource constraints.

Service

3 **Welcome**
6 **News**
96 **Back Issues**
97 **Call for Papers**
98 **Coming Next Month**

Tools

18 **Coroot**
In the deep waters of scalable environments, observability is more important than ever. Coroot enters the scene with eBPF to intercept traffic and collect data that is of interest for monitoring and attack detection.

26 **NAPS2 SDK**
A NAPS2 PDF scanner, the C# language, and the .NET Framework set the stage for a CLI app that scans documents.

30 **Rescue Systems**
Rescue systems play a particularly important role for desktops because they cannot simply be reinstalled on the fly. We look at venerable but outdated Knoppix and three systems still in development: Grml, SystemRescue, and Finnix.

36 **Optimizing File I/O**
File I/O is a limiting factor for many applications and can become a bottleneck for data-intensive computing. We analyze different file access functions and present a script that visualizes and optimizes inefficient file I/O.

Containers and Virtualization

46 **Netdata**
Netdata relies on easy commissioning and a hosted cloud service to simplify monitoring, alerting, and trending. More than 300 services can be monitored in addition to basic system parameters.

54 **dockle**
The open source dockle tool can help keep your Docker containers compliant.

Security

58 **WireGuard Portal**
The wg-portal project helps you set up, manage, and scale a WireGuard VPN environment.

60 **IVRE**
The IVRE tool integrates into existing IT infrastructures for reconnaissance on private networks to help analyze endpoints accessible over the Internet.

@adminmag
@admin-magazine.com
ADMIN magazine
@adminmagazine@hachyderm.io

10 | Lightweight Kubernetes

3 compact Kubernetes compared

K3s, k0s, and MicroK8s vie for performance honors on the control plane and data plane in artificially created extreme stress scenarios.

Highlights

36 Optimizing File I/O
We discuss a Python script that lets you analyze, visualize, and optimize file I/O for more efficient file access patterns in your programs.

60 IVRE
This flexible network recon framework can be integrated into existing IT infrastructures to support both active and passive network detection use cases.

88 AD Risk Mitigation
To operate Active Directory securely, you must have a general plan in place geared toward security and be willing to deviate from the default values.

Management

62 Intro to Software Licenses
Software licenses tell you how you can use software packages and legally deploy the software in your projects. Fortunately, tools can help you automate the process of verifying licenses.

68 Updating with Ansible
Ansible has earned an excellent reputation as a tool for configuration management. Thanks to playbooks, updates are more or less child's play.

Nuts and Bolts

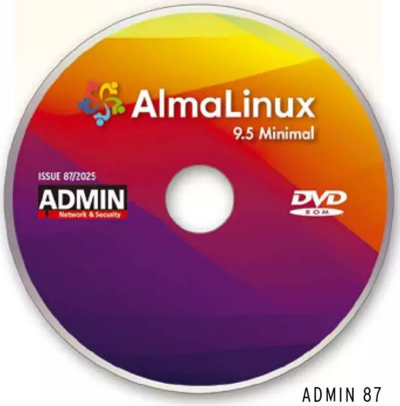
72 Get-NetTCPConnection and ss
This Windows cmdlet and Linux utility are efficient and powerful tools for monitoring network connections and identifying potential security threats on Windows and Linux.

78 AIX Toolbox
Starting your journey with open source software on AIX has never been easier with the IBM AIX Toolbox.

88 AD Risk Mitigation
Active Directory default settings could expose your AD environment to security vulnerabilities. We look at account hygiene and risk mitigation in authentication, login data, PKI, domain join, and more.

On the DVD

AlmaLinux 9.5 Minimal
The Minimal "Teal Serval" release included with this magazine is a self-containing DVD image that can be installed offline. The distribution comes with kernel version 5.14.0-503.x and compiler updates for the GCC (14), LLVM (18.1.8), Rust (1.79.0), and Go (1.22) toolsets, as well as GCC (11.5) and Annobin (12.70). New packages include .NET 9.0 and BIND 9.18. Updates for the PCP and Grafana performance monitoring tool, modifications to device drivers, and numerous security updates round out this offering. For more information, check out the Release Notes [https://wiki.almalinux.org/release-notes/9.5.html#changelog].



News for Admins

Tech News

GitGuardian Introduces NHI Governance

GitGuardian recently announced a new NHI Governance solution, which offers comprehensive lifecycle management of non-human identities (NHIs) to “help enterprises regain control over their secrets and prevent costly breaches.”

NHIs — such as service accounts, API keys, and authentication tokens — “are used as digital references for machine-to-machine access,” explains Soujanya Ain (https://blog.gitguardian.com/gitguardian-launches-nhi-governance/). These accounts differ from human users, which can make them difficult to manage, Ain says. For example, “NHIs lack the additional security layer of multi-factor authentication, and they authenticate using secrets such as API keys, tokens, and certificates — credentials that, once exposed, can be exploited by attackers.”

According to GitGuardian, NHI Governance ensures that NHIs and their secrets are:

- Created securely with least-privilege principles
- Monitored continuously for suspicious access patterns
- Rotated according to compliance requirements
- Revoked immediately when compromised or no longer needed

Learn more at GitGuardian.

IBM Launches LinuxONE 5

IBM has announced IBM LinuxONE Emperor 5 (https://www.ibm.com/products/linuxone-5), a Linux computing platform powered by the IBM Telum II processor with built-in AI acceleration. This release marks the fifth generation of IBM LinuxONE and, according to the company, the system combines “enterprise security, performance and scalability with Linux and open source workloads including hybrid cloud and AI applications.”

LinuxONE 5 aims to help organizations “reduce operational complexity and optimize their IT infrastructure by consolidating workloads across multiple servers onto a single high-capacity system.” For example, the company states, “moving cloud-native, containerized workloads from a compared x86 solution to an IBM LinuxONE 5 running the same software products can save up to 44 percent on the total cost of ownership over five years.”

The available operating systems for LinuxONE 5 are Red Hat Enterprise Linux, SUSE Linux Enterprise Server, and Canonical Ubuntu Server.

Learn more at IBM: https://community.ibm.com/community/user/ibmz-and-linuxone/blogs/tina-tarquinio1/2025/05/06/the-next-generation-of-ibm-linuxone.

OpenSSF Offers Free Course to Help Navigate EU Cyber Resilience Act

The Open Source Security Foundation (OpenSSF), in conjunction with LF Education, is offering a free online course to help software developers understand and prepare for the requirements of the EU Cyber Resilience Act (CRA).

Get the latest IT and HPC news in your inbox

Subscribe free to ADMIN Update
bit.ly/ADMIN-Update

Lead Image © vlastas, 123RF.com

The course — Understanding the EU Cyber Resilience Act (LFEL1001 – https://training.linuxfoundation.org/express-learning/understanding-the-eu-cyber-resilience-act-cra-lfel1001/) — was designed to "address the serious knowledge gap across the software industry when it comes to CRA readiness," says Dr. David A. Wheeler, Director of Open Source Supply Chain Security at OpenSSF.

As the OpenSFF explains, the CRA "introduces cybersecurity obligations for a broad range of software producers — including some open source projects. CRA not only applies to developers in the EU, but to almost every product with software that's distributed in the EU."

The free course — aimed at software decision-makers, developers, and managers — provides information to help participants navigate CRA compliance, mitigate risks, and meet regulatory standards.

Learn more at OpenSSF: https://openssf.org/.

Rapid7 Announces MDR for Enterprise

Rapid7 has launched Managed Detection & Response (MDR) for Enterprise, a customizable detection and response service that combines "deeply tailored detection engineering with full-service 24x7 protection."

MDR for Enterprise (https://www.rapid7.com/services/managed-detection-and-response-mdr/enterprise/) expands the company's MDR offering to provide coverage tailored to each enterprise. Capabilities include:

- Custom event source integration
- Customized detections
- Tailored threat monitoring
- Engagement model and collaboration

MDR for Enterprise also lets you bring your own log sources and extend monitoring into non-standard systems. "Whether it's a homegrown application or a niche vertical-specific tool, we'll build the integration and align detection logic to your context — not the other way around," the company says (https://www.rapid7.com/blog/post/2025/04/24/the-new-rapid7-mdr-for-enterprise-tailored-detection-and-response-for-complex-environments/).

Learn more at Rapid7: https://www.rapid7.com/.

Infoblox and Google Cloud Partner on DNS Security Solutions

Infoblox and Google Cloud have partnered to release two new products aimed at simplifying enterprise networking and security.

Together, the companies are offering the following solutions to help organizations address various challenges in cloud networking and security:

- **Infoblox Universal DDI for Google's Cloud WAN** is a fully integrated solution that provides infrastructure-free DNS and DHCP services, tightly integrated with the Google Cloud Cross-Cloud Network.
- **Google Cloud DNS Armor** leverages Infoblox's protective DNS expertise to secure cloud workloads and provide robust and preemptive detection of malicious activity for Google Cloud workloads.

"Every attack starts with a DNS query," Infoblox states, and "by monitoring these DNS queries, organizations can detect early signs of compromise. If an application unknowingly reaches out to a bad actor-controlled site, it's a clear signal something is amiss."

Learn more at Infoblox (https://blogs.infoblox.com/company/infoblox-collaborates-with-google-cloud-on-protective-dns-security/).

IBM Announces IBM z17 Mainframe Engineered for AI

IBM has announced the IBM z17 mainframe, featuring AI capabilities across hardware, software, and systems operations.

Powered by the new IBM Telum II processor, the IBM z17 is designed to fully integrate into hybrid environments by "joining hardware innovations, software capabilities for AI, and rich support for open standards and tooling."

The system also introduces “multi-model AI capabilities, new security features to protect data, and tools that leverage AI for improving system usability and management,” the announcement states (https://www.prnewswire.com/news-releases/ibm-z17-the-first-mainframe-fully-engineered-for-the-ai-age-302422651.html). Features include a new operating system, the IBM Spyre Accelerator for increased AI compute capabilities, and IBM Z Operations Unite.

The new z/OS 3.2 operating system — expected in the third quarter of 2025 — is “designed to support hardware-accelerated AI capabilities across the system and operational AI insights for system management capabilities. Additionally, z/OS 3.2 will provide support for modern data access methods, NoSQL databases, and hybrid cloud data processing,” the announcement says.

Learn more at IBM: https://research.ibm.com/blog/how-ibm-research-helped-prepare-the-ibm-z17-for-tomorrows-workloads.

2025 Open Source Job Survey Report Released

Open source professionals value professional fulfillment and work-life balance over high wages and benefits when choosing a job role, says the 2025 Open Source Professionals Job Survey Report.

According to the new report from Open Source JobHub (OSJH – https://opensourcejobhub.com/) and Linux Professional Institute (LPI – https://www.lpi.org/), 97 percent of respondents rate professional fulfillment as essential, very important, or somewhat important, giving it the highest ranking of the 10 factors they were asked to consider.

Those working with open source software also value factors such as work-life balance, remote work, open source policy guidelines, and training and certification opportunities.

“This report offers a unique view into what open source professionals are looking for in their job roles,” says Brian Osborn, Founder of OSJH, and CEO and Publisher at Linux New Media. “Understanding the perspective of these admins, developers, and non-tech pros is key to finding and retaining open source talent.”

Complete survey results are available now in the free 2025 Open Source Professionals Job Survey Report: https://opensourcejobhub.com/blog/2025-open-source-professionals-job-survey-report.

GitHub Launches Free Secret Risk Assessment Tool

GitHub has launched a free secret risk assessment tool, as part of recent changes to its Advanced Security offerings.

The free tool, which is now available in the Security tab, “gives admins and developers a clear view of where secrets are exposed across their organization” and offers actionable steps to better secure their environments.

The company has also unbundled GitHub Advanced Security (https://github.com/security/advanced-security) to make the following tools available as standalone products:

- **GitHub Secret Protection:** Detects and prevents secret leaks before they happen using push protection, secret scanning, AI-powered detection with low false-positive rate, security insights, and more. Available at $19/month per active committer.
- **GitHub Code Security:** Helps identify and remediate vulnerabilities faster with code scanning, Copilot Autofix, security campaigns, Dependency Review Action, and more. Available at $30/month per active committer.

GitHub Team (https://github.com/team) customers can purchase these products without a GitHub Enterprise subscription.

Learn more at GitHub: https://github.blog/security/application-security/next-evolution-github-advanced-security/.

DigitalOcean Kubernetes Service Doubles Cluster Capacity

DigitalOcean has announced a major update to its DigitalOcean Kubernetes Service (DOKS), which now allows organizations to scale up to 1,000 nodes per cluster.

Key features of DOKS include:

- Increased cluster capacity from 500 to 1,000 worker nodes per cluster
- Optimized networking with VPC-native Kubernetes
- Improved performance with eBPF-powered networking
- Managed Cilium for high-performance networking

“This expansion enables organizations to seamlessly run larger workloads within a single cluster, simplifies secure cloud networking, and significantly reduces operational overhead,” said Bratin Saha, Chief Product and Technology Officer, DigitalOcean.

Read more at DigitalOcean: https://www.digitalocean.com/blog/introducing-doks-1k-nodes.

Sonatype Offers End-to-End AI Software Composition Analysis

Sonatype has announced AI Software Composition Analysis (AI SCA – https://www.sonatype.com/solutions/open-source-ai), which extends the company’s platform to provide “instant visibility, policy, and control over your AI/ML usage in your applications and data pipelines.” AI SCA offers comprehensive AI model governance, including automated reports on AI/ML usage.

“It has never been easier for organizations to integrate open source AI models into software, but with open source AI consumption comes the same risk facing users of traditional open source. It is imperative that we, as an industry, secure their use now in order to prevent unmanageable security workloads in the future,” said Brian Fox, co-founder and CTO at Sonatype.

To help organizations manage open source AI/ML usage in software supply chains, the end-to-end solution provides:

- Proactive AI threat detection
- Centralized AI model governance
- Automated AI policy management
- AI observability and compliance

Learn more at Sonatype: https://www.sonatype.com/.

Unmanaged Open Source Components Pose Serious Risks, Says Black Duck Report

Open source software offers many benefits but, without proper management, can also involve “widespread vulnerabilities, license conflicts, and maintenance challenges,” according to Black Duck Software’s 2025 Open Source Security and Risk Analysis (OSSRA) report.

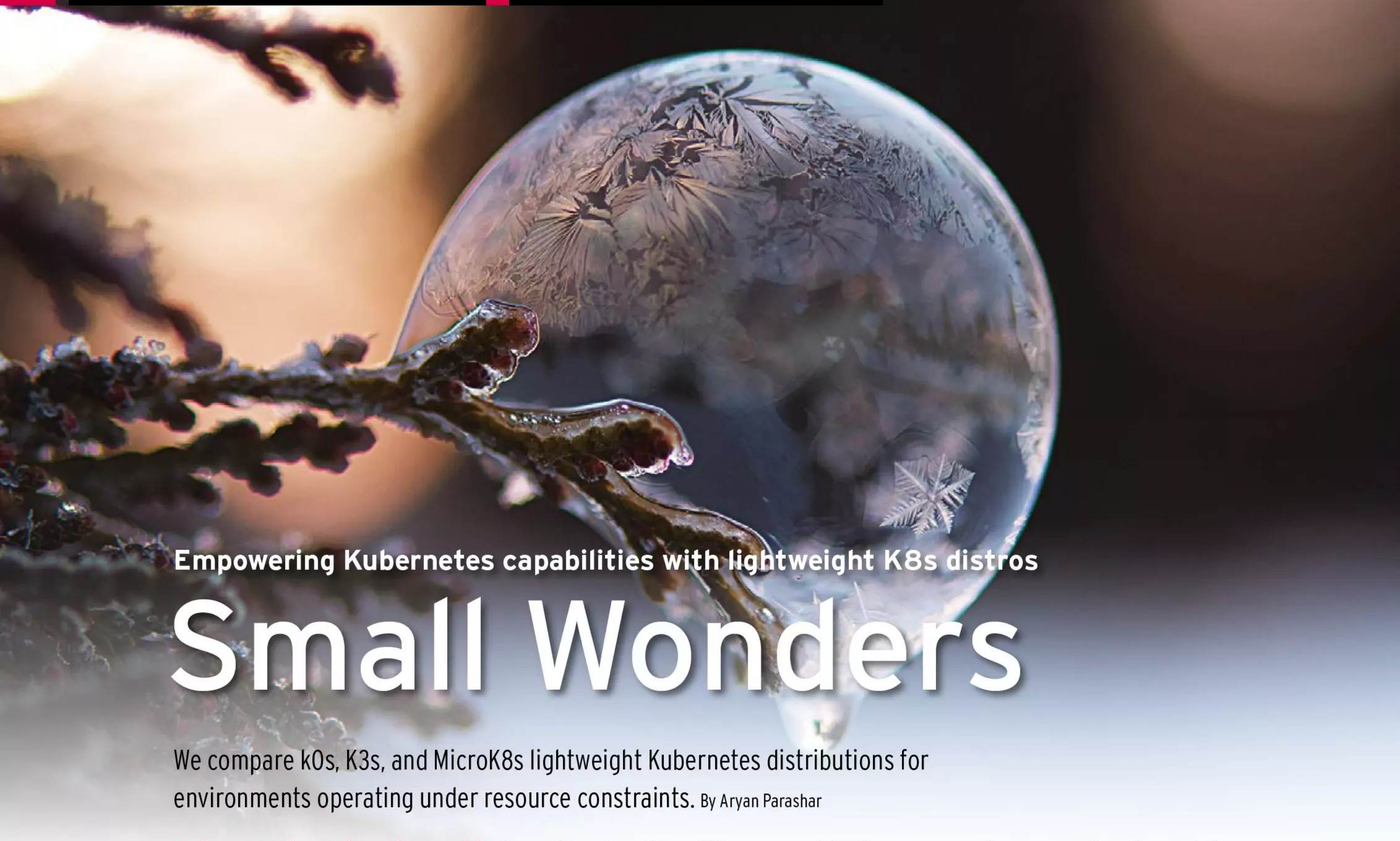
The latest report “highlights the pervasive nature of open source software and the significant risks associated with its unmanaged use,” said Fred Bals in a related blog post (https://www.blackduck.com/blog/open-source-trends-ossra-report.html).

Top findings from this year’s report include:

- Ninety-seven percent of all applications evaluated for the report contain open source.
- Ninety percent of audited codebases had open source components more than four years out of date.
- Eighty-six percent of commercial codebases evaluated contained open source software vulnerabilities.
- Eighty-one percent of these codebases contained high- or critical-risk vulnerabilities.
- Fifty-six percent of all audited applications had license conflicts.

“The core message of OSSRA 2025 is that organizations must have comprehensive visibility into their code, proactively manage open source risk, and adopt robust security and compliance practices,” Bals said.

Read the full report at Black Duck: https://www.blackduck.com/resources/analyst-reports/open-source-security-risk-analysis.html.



Empowering Kubernetes capabilities with lightweight K8s distros

Small Wonders

We compare k0s, K3s, and MicroK8s lightweight Kubernetes distributions for environments operating under resource constraints. By Aryan Parashar

Kubernetes has taken the world of application infrastructure by storm. Whether you're working with microservices, scaling cloud-native apps, or deploying containers, Kubernetes has become the go-to solution. Its flexibility and scalability make it a powerhouse, but honestly, it can be pretty heavy and complex. Lightweight Kubernetes distros like k0s, K3s, and MicroK8s offer a more streamlined and resource-efficient way to enjoy the power of Kubernetes without all the bulk.

Many people argue that lightweight Kubernetes distros might lose their edge when faced with high-overhead computations, but as the saying goes, the proof is in the pudding. The real magic lies in testing to uncover the truth behind this debate and how they stack up against each other. In this article, I dive into some key differences and features to help you understand how each distro stacks up, from deployment models to CPU architecture support.

Cutting the Fat

When tackling resource constraints in a small-scale development environment, a single computer can handle just a few tricks when sampling the landscape of services, and you only get a taste of some integrations and functionalities. When passed to the development environment for testing and integration, it gets even more complex, resulting in unit tests slowing down and having to wait for the environment to come up, which does not allow continuous integration and continuous deployment (CI/CD).

In these cases, the need for a lightweight distribution that can work on personal laptops, systems, or even server-side machines that live on the edge comes into focus with stripped down versions of Kubernetes that have modest resource consumption but similar or even more Kubernetes functionalities. For CI/CD pipelines, these lightweight Kubernetes distributions will be a lot more efficient because of their modest resource consumption and faster response times, computation, and installation.

When it comes to Kubernetes distributions like k0s, K3s, and MicroK8s, versatility is the name of the game. These lightweight versions are so flexible that you can run them on something as small as a Raspberry Pi, all the way up to edge locations or even a single rack in a data center. They're designed to handle a wide range of environments without breaking a sweat. Whether you're tinkering with home labs or managing industrial edge deployments, these distros give you the power of a slimmed-down version of Kubernetes (K8s), ready to take on whatever you throw its way, without all the extra weight.

A Closer Look

In the world of container orchestration, full-scale Kubernetes can be overkill for smaller setups, edge computing, or resource-limited environments. The lightweight distributions maintain the Kubernetes's core functionality while being more resource efficient, easier to deploy, and highly flexible across different architectures and operating systems.

As Table 1 shows, one of the biggest advantages of lightweight Kubernetes distributions is how easy they are to set up and run. However, the installation processes differ slightly depending on the distribution.

Single-Binary Installation

One of the best things about k0s [1] and K3s [2] is that they are super lightweight and designed for quick, no-fuss installations. Both of these distros can be installed as a single binary, meaning you can just download one file and you're good to go.

If you're installing k0s or K3s, it's as simple as running commands that

Photo by Aaron Burden on Unsplash

Table 1: Key Architectural Differences

	k0s	K3s	Micro K8s
Main developer	Mirantis	Rancher/SUSE	Canonical
Programming language	Go	Go	Python, Shell
Operating systems	Linux, Microsoft Server (2019 exp.)	Linux	Ubuntu (default), Linux, Windows, macOS
CPU architectures	x86-64, ARM64, ARMv7	x86, ARM64, ARMhf	x86, ARM64, s390x, Power9
Deployment	Single binary	Single binary	Snap package
FIPS 140-2 accreditation	Yes	No	No

download the binary file, set up the required services, and start the cluster – all in just a few steps:

```
curl -sSLf https://get.k0s.sh | ↩
  sudo sh
curl -sfL https://get.k3s.io | sh -
```

The beauty of this process is that it's incredibly lightweight, with no need to mess around with complex package managers or dependencies for quick and efficient installation that is perfect for edge devices, Internet of Things (IoT) setups, or environments where resources are limited. You essentially get the full power of Kubernetes, but without the bloat.

Snap Package Installation

MicroK8s [3] takes a slightly different approach. Although it's also a lightweight Kubernetes distribution, it is packaged and distributed as a snap package. If you're using Ubuntu, the process is straightforward, because snap is native to Ubuntu. However, if you're on other Linux distros, you will need to make sure `snap` is installed first. To install MicroK8s, run:

```
sudo snap install microk8s --classic
```

The snap package bundles everything you need into one package, much like a container for the software. Snap is designed to work across various Linux distributions and ensures that everything, including dependencies, is contained in the package itself. The key difference is that snap packages are sandboxed from the rest of your system, meaning they operate in isolation. This arrangement gives MicroK8s a bit more control over its environment, but it also means slightly more overhead compared with just downloading a binary, as for k0s or K3s. Additionally, snap packages need `snapd` (the snap package manager) installed and running in the background.

k0s

Figure 1 shows how the k0s lightweight Kubernetes distribution packages its core components to keep things simple and efficient. This all-in-one approach to Kubernetes has everything you need for a fully functional Kubernetes cluster, neatly bundled into the single `k0s.bin` binary, without unnecessary overhead. At its heart is the *k0s (logic)* area, which is where all the Kubernetes magic happens. Inside this block are the usual suspects, including `kubelet`, `containerd`, and all the key Kubernetes components like the API server, scheduler, and controller manager. These are the brains behind your cluster responsible for managing workloads, orchestrating containers, and ensuring everything runs smoothly. However, k0s doesn't stop there. It also includes components like `runc`, which is essential for running containers, and Kine, a lightweight data store that acts as a replacement for etcd in smaller setups. Security enthusiasts get FIPS 140-2 accreditation, which means it meets certain cryptographic standards – a big deal for highly regulated environments.

The second section in Figure 1, labeled *images (for airgap)*, ensures that k0s can be deployed in airgapped environments. Airgaps are development setups in which a cluster is completely isolated from external networks, including the Internet, which is normally in Kubernetes nodes because of security measures. This arrangement makes it super

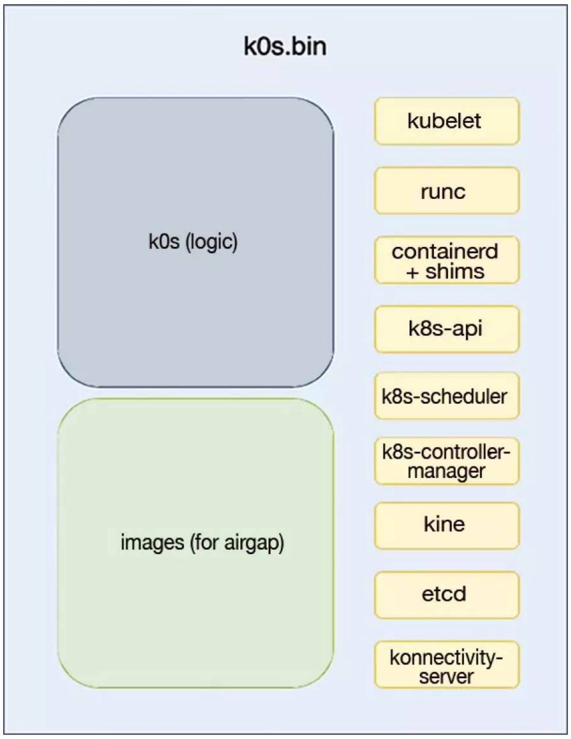


Figure 1: Unpacking the k0s architecture. © Mirantis Inc. k0s project [4]

versatile for use cases that need tight security or are deploying in environments with limited connectivity.

What makes k0s stand out is its simplicity. All of these crucial components are prepackaged into one binary, so you're not juggling multiple pieces of software. It's like having a Kubernetes cluster in a tidy little box, ready to go wherever you need it.

Because the cluster can't pull images from public Docker registries, a private container registry is maintained to store and provide necessary container images; thus, the operating system and k0s can be upgraded separately, such that if any security updates exist, you can just download them on the k0s binary, and everything is updated independently.

Lifecycle Management

Managing a multinode Kubernetes cluster doesn't have to be a headache, especially with tools like `k0s controller` and `k0sctl`. These handy tools make your life a lot easier by handling everything from installation and updates to backups, restores, and even cluster resets – all with just a few commands. It's a full toolkit for managing your entire k0s cluster without breaking a sweat.

Even better, you can deploy individual machines into your k0s cluster without manually setting up each node individually. Thanks to `k0smotron`, a Cluster API provider for k0s, this process is fully automated. With these tools, you can easily scale your environment by adding machines dynamically, without the need to configure each node from scratch. Just point and click, and your cluster grows or shrinks as needed.

This level of automation makes k0s super scalable and incredibly easy to manage, whether you're running it on premises or in the cloud. No more juggling complex scripts or configurations: `k0smotron` does the heavy lifting for you.

You can also run your workloads on control nodes unless `taint` and `tolerations` are applied to prevent certain conditions that pose a resource consumption hit between the critical control plane components and regular workloads that could affect cluster efficiency and stability.

k0s Approach

k0s follows a design in which only the management (control plane) components are running on control planes and no workloads are scheduled on these nodes by default. Figure 2 shows how k0s ensures a clear separation of roles, leading to a more stable and predictable control plane. This method guarantees the workload wouldn't be accidently scheduled on control nodes and that the API server, controller manager, and other components always have resources they need without sharing. The figure also illustrates how keeping workloads away from nodes where cluster management and configuration operations take place reduces potential attack vectors.

k0s-in-a-Pod

One of the coolest features of k0s is its ability to run inside a Kubernetes cluster, known as k0s-in-a-pod. In this scenario, you run a lightweight k0s distribution as a set of pods inside another Kubernetes environment, effectively creating Kubernetes-in-Kubernetes. In this way, you can spin up entire Kubernetes clusters within your existing cluster. These virtual clusters are fully functional, with their own API server, controller manager, scheduler, and so on, running independently but sharing resources with the host cluster.

Creating isolated Kubernetes clusters for different teams or projects without provisioning new hardware is a game-changer for multitenant environments. It's also perfect for testing or CI/CD pipelines, letting you spin up temporary clusters on the fly and tear them down just as easily. Because k0s is lightweight, it doesn't bog down your resources.

With the use of tools like `k0smotron` (the Cluster API provider for k0s), you can automate the deployment and management of these virtual k0s clusters, turning complex multicluster management into a seamless process. Whether you're scaling infrastructure or testing new setups, k0s-in-a-pod makes Kubernetes even more flexible and easy to manage.

Once your k0s cluster is deployed inside Kubernetes, you can interact with it just like any other Kubernetes cluster. You can get the `kubeconfig` file to access this virtual cluster with:

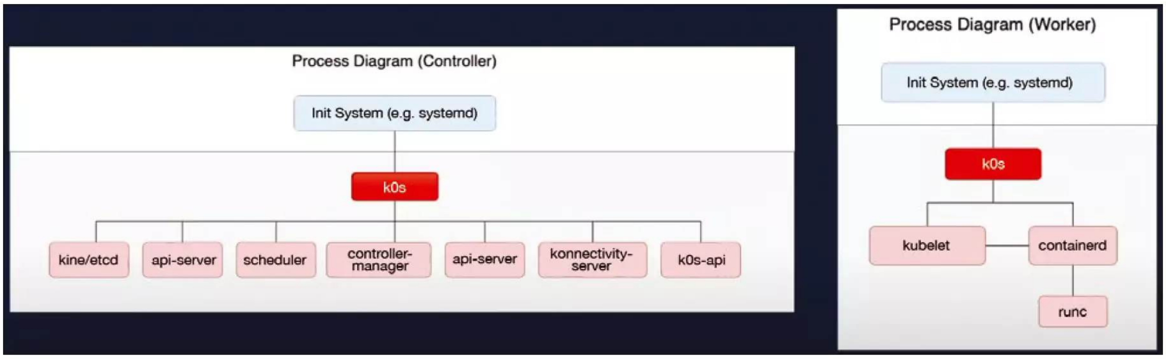


Figure 2: Process diagrams for k0s: The left diagram illustrates the process architecture of the k0s controller, and the right diagram shows the k0s worker node architecture. © Mirantis Inc. k0s project [4]

```
kubectl get secret my-virtual-cluster ⮐
  -n k0smotron ⮐
  -o jsonpath="{.data.kubeconfig}" | ⮐
  base64 --decode > k0s-kubeconfig.yaml
```

Then, use the `kubeconfig` file in this way

```
kubectl --kubeconfig=k0s-kubeconfig.yaml ⮐
  get nodes
```

to interact with your virtual k0s cluster.

No External etcd

k0s does not use an external etcd by default; instead, it uses SQLite for storing the cluster state, which makes the control plane much easier to operate and lighter weight for smaller clusters or edge environments.

For larger clusters you can still opt to use etcd, but SQLite inclusion means easier management for smaller use cases in which full-blown etcd clusters might be overkill. The advantages include:

1. Fewer operational requirements, which means no need to manage a separate etcd cluster unless necessary.
2. Lightweight SQLite, which doesn't require the same level of maintenance as does etcd, simplifying the setup and making it appealing for edge computing environments.
3. Advantages in edge computing environments because of it's lightweight K8s operation, particularly for IoT devices and other resource-constrained setups.

Traditional Kubernetes clusters used to have a significant resource overhead with a control plane running several components independently. k0s minimizes overhead by tightly packing everything in a compact form, making it more efficient for edge computing.

Control Plane and Security

k0s optimizes control plane isolation by completely disallowing workloads on the management nodes and by limiting the system services running on them. This clear separation and streamlined connectivity make k0s a strong candidate for deployments where minimalism, security, and ease of use are key, particularly in edge computing or IoT environments. Security is handled by k0s as follows:

Bare-bones management. Only the essential control plane components run on the management (control plane) nodes. This minimalistic approach focuses solely on running core control plane services, which are necessary to manage the cluster. This term also refers to how k0s runs as a simple Linux process that does not rely on extra system services like `containerd` or `kubelet` on the control plane, which keeps the setup lightweight and ensures that the control plane isn't burdened with unnecessary background services. Streamlining means only the absolutely necessary components are running, reducing both complexity and the resource footprint.

Simplified connectivity between control plane and worker nodes. Traditional Kubernetes clusters might require that several different ports be open for communication between control plane nodes and worker nodes, which increases the complexity of network configuration and the attack surface. k0s simplifies with: (1) Single-port communication, wherein k0s, the control plane, and worker nodes communicate through a single port, significantly simplifying network management and reducing the need to configure multiple firewall rules. (2) A lightweight connectivity agent that handles communication between the control plane and worker nodes in k0s and takes care of establishing and maintaining the connection between these components without relying on heavy or complex networking configurations.

Security improvements. k0s inherently improves security by reducing the number of open potential entry points for attackers to just one, thus reducing the attack surface, which is particularly important in edge computing and IoT scenarios, where clusters might be deployed in less secure environments and exposed to the public Internet. You only need to manage rules for one port, which decreases administrative overhead and reduces the chance of misconfigured network security settings, thus simplifying the firewalls rules.

To verify the security ports used and check whether k0s is minimizing the number of open ports, you can use a few commands (**Listing 1**) that inspect the network and firewall configuration on your system. These commands can help you confirm whether only one port is open and whether network traffic is being managed properly. To check the open ports on your system, you can display a list of open TCP and UDP ports and inspect the configuration to verify which ports are explicitly open. For additional verification, you can use tools like `nmap` to scan for open ports. Also, you can ensure your firewall configuration is properly configured by current firewall rules to allow only the necessary ports for k0s.

Control and Data Plane Performance Comparison

Research at ABB Corporate Research [5] on lightweight distributions like k0s, K3s, MicroK8s, and MicroShift looked at the computational behavior in different situations for different processes with the K-Bench benchmark (**Figure 3**). The upper row shows the throughputs for individual pod creations (i.e., eight pods in

Listing 1: Verifying Security

```
# Look for k0s-related ports to ensure only one is in use
sudo ss -tuln

# Confirm that only the required port (typically the API
  port) is open
sudo iptables -L -n -v

# Determine open and blocked ports
sudo ufw status

# Verify open ports
cat /etc/k0s/k0s.yaml | grep port

# Scan for open ports
sudo nmap -p- localhost
```

parallel), as well as the pod creation average latency. This value shouldn't be confused with pod startup latency, because here it is computed by subtracting the pod creation timestamp from the first timestamp of the scheduling event associated with the pod, which excludes the phase after pod creation before the pod's status changes to *Running*. K3s marginally shows the highest throughput and lowest latency, although MicroK8s and k0s are not far off (about 15 percent difference). The results for MicroShift are not directly comparable because the pods were created on a single node and could not be distributed in a cluster of three nodes. The lower row shows the same metrics for starting a deployment of 40 pods (i.e., around 13 pods per node). Again, K3s barely shows the highest throughput and lowest latency, although the differences are even less pronounced. Compared with the former eight-pod creation, the latencies go up from 2-4 seconds to 9-18 seconds for all distributions, indicating the aforementioned queueing effects.

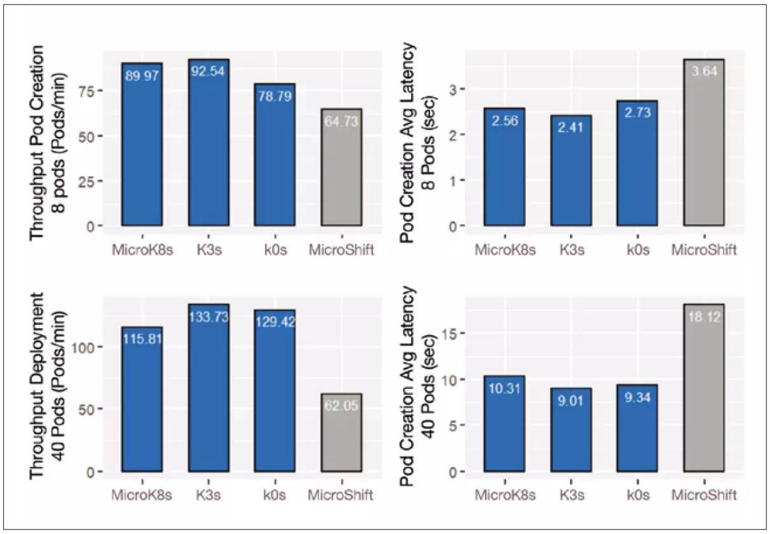


Figure 3: Pod creation throughput and latencies (eight clients). MicroShift was tested on a single combined controller/worker node; therefore, the measurements are not directly comparable to the others.

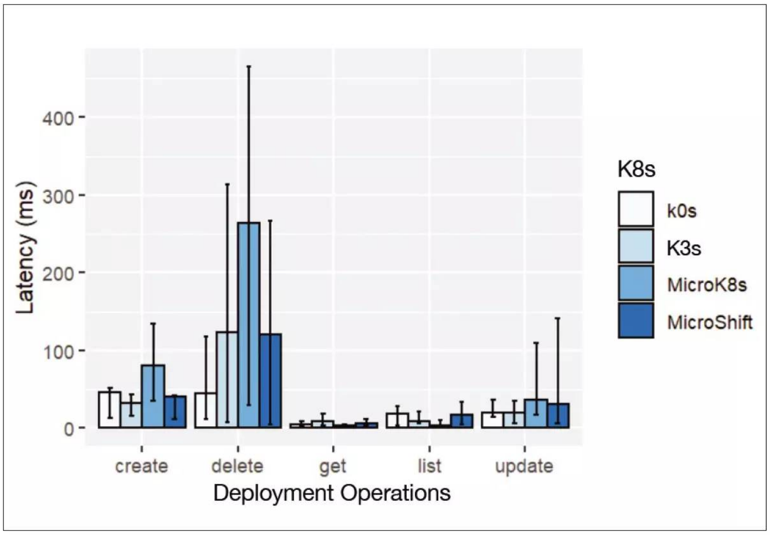


Figure 4: Latency (minimum, median, and maximum) during deployment operations (eight clients, 40 pods).

Figure 4 visualizes latencies when creating, deleting, getting, listing, and updating a K8s deployment resource. K-Bench reports just the minimum, median, and maximum values. These operations only cause data operations on the controller nodes and are independent of the number of nodes used. Here, the MicroShift results are meaningful. The `create` and `delete` operations by far show the highest latencies, whereas the `get`, `list`, and `update` operations are much shorter. MicroK8s shows the highest latencies, with `delete` operations going up to a maximum of 470ms. The differences are likely caused by the different control plane data stores (e.g., MicroK8s uses Dqlite and K3s uses SQLite). k0s was significantly faster in deleting a deployment compared with the other distributions.

The results were extended by running a workload three times as heavy to determine how the throughputs and latencies were affected. The results in Figure 5 are from a separate series of K-Bench experiments, now starting 24 pods in parallel, as well as starting a more extreme scenario of 120 pods in parallel (i.e., 40 pods per node). The pod creation throughput increased to a maximum of 152 pods/min (k0s), with pod creation latency increasing up to 37.7 seconds (MicroK8s). K3s and k0s show significantly higher throughputs and lower latencies than MicroK8s.

The latter experiments also recorded the pod operation latencies (Figure 6). Under the higher workload, the latencies are significantly higher than for the deployment operations before, but a similar profile again is visible, with MicroK8s showing the longest creation and deletion latencies and k0s and MicroShift having the shortest latencies.

All of the operations (`create/delete/get/list/update`) coarsely lead to the same ranking of the lightweight K8s distributions. MicroK8s shows the longest latencies for the different operations. Its Dqlite database is a distributed version of SQLite that can

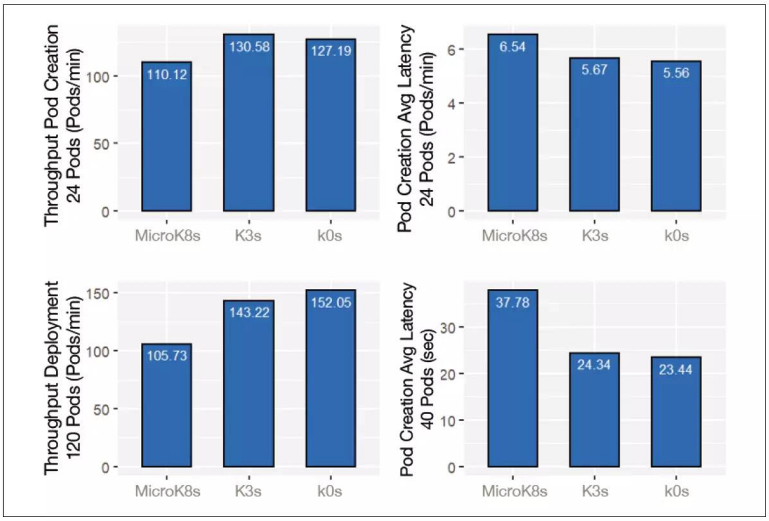


Figure 5: **Performance comparison of lightweight Kubernetes distributions (MicroK8s, K3s, and k0s) in terms of pod creation and deployment. K3s and k0s demonstrate higher throughput and lower latency compared with MicroK8s.**

scale horizontally and uses C-Raft – an optimized Raft implementation in C – as the consensus algorithm. It could be that the longer latencies for MicroK8s are related to the Raft algorithm that ensures data consistency in a cluster, whereas SQLite might not need any locking in this scenario with a single controller node.

Data Plane Performance

Results from the `memtier_benchmark` utility from the data plane experiments compared with K-Bench showed MicroShift had the lowest latencies and the highest throughput (Figure 7). The average latency for the database operations lies between 10 and 19ms on average, with few outliers going up to 174ms. MicroShift was the fastest distribution, showing average latencies 15 percent shorter than MicroK8s and 38 percent shorter than K3s. The achieved throughput is a consequence of the latencies, so MicroShift also has the highest numbers for throughput operations.

The data plane performance experiments used the same container (NGINX), same application (memtier), same workload, and same hardware. Because these experiments hardly had any control plane operations, a major factor contributing to the results is the container runtime. MicroShift uses CRI-O, whereas the other distributions use `containerd`. MicroShift executed on Red Hat Enterprise Linux (RHEL) instead of Ubuntu, which could also influence the results. Furthermore, as seen in Figure 3, each distribution has a considerable CPU overhead, even when idle, which might explain the comparably poor data plane performance results for K3s, which also had the highest controller CPU utilization in the idle experiments.

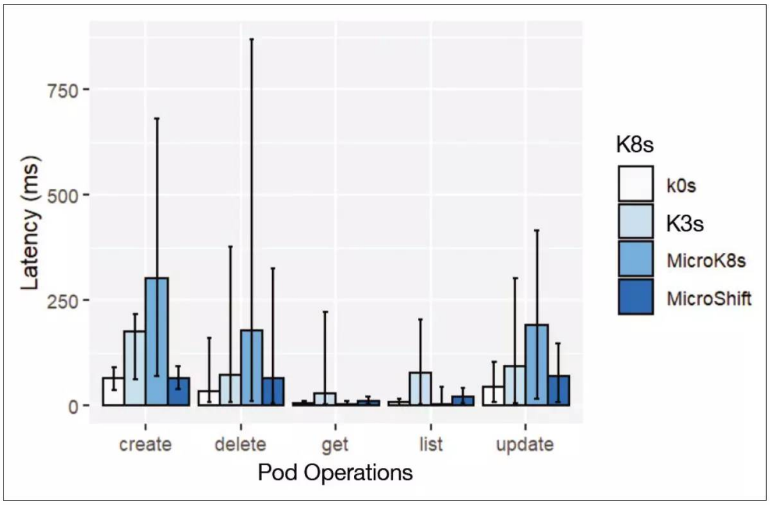


Figure 6: **Latency behavior of different Kubernetes distributions with different `create`, `delete`, `get`, `list`, and update operations.**

Decision Supports

MicroK8s was initially meant for developer workstations and was only subsequently optimized further for a low-memory footprint. It favors easy extensibility over minimal resource usage and may be a preferred choice when using Ubuntu and the snap package manager, although it also can be installed in Windows. High availability in MicroK8s is activated automatically on clusters with three or more nodes. Patch release updates are installed automatically in MicroK8s. The tests confirmed the convenient activation of prebuilt add-ons (e.g., dashboard, Prometheus) with single commands, which in contrast typically requires a number of configuration steps for the other distributions. **K3s** has been designed to reduce binary size (64.5MB) and memory footprint targeting resource-constrained environments such as edge clusters as well as developer workstations. Memory footprint was reduced by combining all components into a single process, and binary size was reduced by removing third-party storage drivers and cloud providers, for example, from K8s. Updates follow the upstream K8s releases, with a goal of patch release within one week and new minor releases within 30 days. K3s has found widespread popularity because of its light weight and easy install

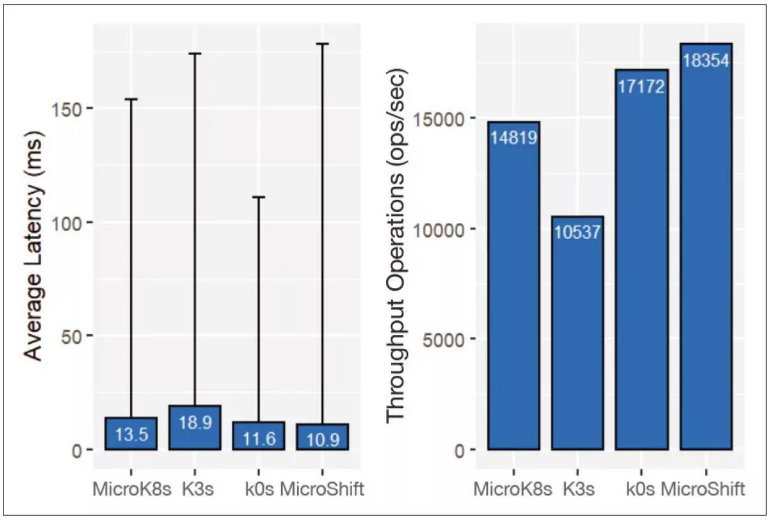


Figure 7: **Performance comparison results collected by memtier for average latency and throughput operations of all Kubernetes distributions (k0s, K3s, MicroShift, MicroK8s).**

procedure, with only most essential components included, and it could be a preferred choice for resource-constrained devices, especially if those are dedicated worker nodes.

k0s appears similar to K3s and is also rigorously tuned for low resource usage and easy installation. It has so far only had low popularity. The developers highlight that the design was done with security in mind, allowing 100 percent FIPS compliance if a proper toolchain is in place. Because the k0s binary is self-contained, security vulnerabilities can potentially be fixed quickly, not being dependent on external components. k0s intentionally reduces bundled add-ons, such as ingresses, service meshes, and storage, because they follow opinionated design patterns, making them less flexible and harder to maintain in some scenarios. In these tests, k0s appears to be lean and streamlined, with a very easy install procedure.

Conclusions

The tests showed marginal differences in control plane performance between the lightweight K8s distributions, with MicroK8s having slightly poorer performance than K3s and k0s, which performed almost identically in pod throughput and pod creation latency. The performance was measured for the default configurations of the distributions, although default Metrics Servers were deactivated for all and k0s was reconfigured to allow combined controller worker nodes to achieve a fair comparison. Further performance optimization might be possible by reconfiguring the K8s distributions or integrating custom components through their different extension mechanisms.

Control plane tests were performed on artificially created extreme stress scenarios, which might be rare in day-to-day practice, especially on resource-constrained edge devices. For more modest, regular workloads, you might not perceive significant performance differences among the K8s distribution used. However, the results here confirmed that the creation of more than 100 pods is feasible on a single node for all distributions, which implies that the K8s distributions set no limits on possible workloads when constrained by the available hardware.

Data plane performance showed some more significant differences between the distributions, with K3s having poor performance on the combined controller/worker node in this scenario. This result seems to correlate with the high controller resource usage of K3s in idle conditions. Possibly the continuous resource usage of the control plane components decreased the achievable throughput in this scenario. However, in practice, such high-load scenarios might use dedicated worker nodes, which would be released from control plane resource usage. On some standalone edge devices with challenging machine learning workloads, the data plane performance could be relevant. However, when the lightweight K8s distributions are used on powerful developer workstation, it should be less of a problem. The results also indicated that the data plane performance is very similar between MicroK8s and K3s; however, they did not fully exhaust the available resources.

Get your hands dirty by trying out each distribution (Table 2). To know more about the Kubernetes distributions and their importance in the cloud-native landscape, head on over to the official Kubernetes documentation [6]. ■

Info

[1] k0s: [https://k0sproject.io/]
[2] K3s: [https://k3s.io/]
[3] MicroK8s: [https://microk8s.io/]
[4] k0s architecture: [https://docs.k0sproject.io/v1.23.5+k0s.0/architecture/?h=architecture]
[5] Koziolek, H., and N. Eskandani. "Lightweight Kubernetes distributions: A performance comparison of MicroK8s, K3s, k0s, and MicroShift." *In:* ICPE'23. *Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering* (Association for Computing Machinery, 2023), pg. 17-29, [https://dl.acm.org/doi/10.1145/3578244.3583737]
[6] Kubernetes distro partners: [https://kubernetes.io/docs/home/]

Author

Aryan Parashar is Developer Advocate and Ambassador at Cloud Native Computing Foundation (CNCF) landscape projects and creates YouTube videos about cloud-native WebAssembly (Wasm), sustainability, machine learning (ML)Ops, large language model (LLM)Ops, and open source. You can find his content at [https://www.youtube.com/@AryanParashar_].

Table 2: Performance and Architecture Comparison

Feature or Aspect	k0s	K3s	MicroK8s
Control plane architecture	Bare-bones control plane, only core services (API server, scheduler, controller manager). No workloads allowed on control nodes.	Lightweight but includes additional system services like `containerd`. Workloads can be scheduled on control nodes unless restricted by `taints` or `tolerations`.	Modular traditional control plane. Control plane and workloads can run together. Flexible optimization for different environments.
Management processes	Runs as a simple Linux process, with no unnecessary services. Highly efficient in terms of resources.	More system services are bundled by default, leading to a slightly larger footprint than k0s.	Uses containerized components; flexible for different environments and scales well in edge or cloud environments.
Workload on the control plane	Not allowed. Strict separation of control plane and workloads for enhanced reliability.	Allowed unless specifically restricted by `taints` or `tolerations`, meaning less strict separation of workloads and management.	Can run workloads on control nodes, which is especially useful in single-node setups. Makes it more flexible but can consume resources meant for control plane management.
Minimum CPU cores required	1	1	2
Minimum RAM required (GB)	1	1	4
Total memory consumption (MB)	510	512	540
Security	Secure by design with minimized attack surface. Only one port is required for communication between control and worker nodes, which simplifies firewall rules and reduces potential vulnerabilities.	Security depends on the configuration. Requires opening multiple ports for different services, increasing potential attack surface if not properly secured.	Offers flexible security depending on the setup, but a larger set of services and ports might be open depending on enabled components (e.g., network plugins).
Datastore uses	SQLite as the default datastore, which can be scaled to etcd for high-availability setups. Keeps the data layer simple in low-resource environments.	SQLite as the default datastore with an option to use etcd for high availability, but allows workloads on the control plane.	By default, can use Dqlite, which is a Raft-based lightweight database, or etcd when high availability is required. Like K3s, workloads can run on the control plane.
Networking (CNI)	Simple network configuration with one port required for node communication, reducing the need for complex firewall rules.	Slightly more complex networking, requiring more than one port for various services depending on the setup.	Offers multiple container network interfaces (CNIs; Calico, Flannel), with more flexibility in networking, but requires configuring ports on the basis of the selected CNI solution.
CNI interface	k0s doesn't bundle in the default CNI plugin, giving you more flexibility in choosing your networking setup. Users can add Calico, Cilium, or others as needed.	Uses Flannel as the default network back end, which is sufficient for most lightweight deployments but might lack the advanced features found in more complex network solutions like Calico.	Provides built-in support for Calico and Flannel as networking solutions. Calico is more feature rich and supports network policies, whereas Flannel is lighter and easier to configure for simpler use cases.
Container runtime	Delegates runtime management to the workers, focusing solely on control components. Uses a connectivity agent for communication.	Bundled `containerd` runtime by default, which increases the size and footprint of the control plane.	Uses `containerd` as the default runtime, with the ability to enable or disable depending on the setup. Containerized control plane components increase flexibility but could increase complexity.
Deployment use cases	Best suited for environments where strict separation between control and workloads is needed and resource efficiency is critical (e.g., edge, IoT, or secure deployments).	Suited for resource-constrained environments where some flexibility between control and workload nodes is acceptable. Great for edge devices and small systems like Raspberry Pi.	Ideal for single-node, multinode, and cloud setups with flexible component selection and scaling options. Can be used in both small and production-grade clusters.

■

Intelligent observability with AI and Coroot

Silent Observer

In the deep waters of scalable environments, observability is more important than ever. Coroot enters the scene with eBPF to intercept traffic and collect data that is of interest for monitoring and attack detection.

By Martin Loschwitz

Intrusion detection and intrusion prevention are part of everyday life for admins. Solutions for these issues are basically very straightforward beasts that analyze data streams and use sophisticated heuristic methods to identify patterns that point to an imminent or ongoing attack. If any suspicious activity is noticed, an alarm helps you launch countermeasures at an early stage.

The developers at Coroot [1] go beyond the usual intrusion detection systems (IDS) on the market today and offer a kind of artificial intelligence (AI) IDS as a service, revealing several interesting technical details that are worth a closer look.

Dumb Systems

The countermeasures of solutions typically found on the market today can be anything from simple packet filters to sophisticated firewall constructs. Firewalls, for example, leverage connection states as a parameter to decide whether or not to deliver packets. Today's IDS or intrusion prevention systems (IPS) no longer simply police intrusions, they often offer options that automatically identify and warn you about distributed denial of service (DDoS) attacks so that you can immediately block the attack at the network level.

Although these solutions are anything but easy to use because they require a high level of skill and familiarity, these systems are in fact "dumb" in and of themselves. At the end of the day, they just use predefined patterns to determine whether or not an attack is taking place. The principle is similar to that of antivirus software, which only detects malware if it finds a matching signature in its database; it needs a description of the files and checksums that comes with the virus. If this signature is missing for some reason, the program is blind to the threat; thus, a lucrative malware market thrives on the darknet, where previously unknown exploits are sold over the virtual counter for large mounts of money.

AI Improvements

From today's perspective, classic anti-spam tools can be regarded as a first tentative approach toward automatic pattern-based prevention of malicious activities on the Internet. SpamAssassin [2] and Rspamd [3], to name just two examples, rely on heuristic methods to analyze various factors under the hood before deciding whether a message is spam.

These methods improve with user training and regular corrective intervention as follows: If email from a Nigerian prince with an unlikely offer of $20 million is spam, then the same conclusion probably applies to an email from a bank in Guinea with the same offer. A number of factors play a role, including, for example, the probability of such an offer from a stranger reaching the recipient by email completely out of the blue.

No matter how you look at it, these systems have nothing to do with AI; they work purely on the basis of manual training with a few automated features.

Photo by Gustavo Xavier on Unsplash

In the wake of the AI wave currently sweeping the industry, resourceful companies are promising better spam filters and better AI-based intrusion IDS and IPS. As a reminder, the vast majority of AI incarnations are primarily based on large language models (LLMs; i.e., very sophisticated statistics). The principle always remains the same and is very similar to the heuristics of spam filters: From an arbitrary amount of training data, the system is taught to respond to certain events in a certain way.

The main difference between AI models and tools such as SpamAssassin is the use of unsupervised learning that ultimately aims to train the models without human intervention. Unsurprisingly, IT now has a widespread desire to use AI to detect attacks and intrusions, which has been covered several times in this magazine in the past, one example being the various AI plugins for Prometheus [4]. These plugins detect specific patterns in metrics data automatically with the use of AI libraries, which in turn is what enables Prometheus to trigger alarms.

Coroot

Up front, Coroot is a centrally hosted service in the cloud with a subscription-based business model. Although the API always needs to run in the local setup, the AI capabilities are stored in the cloud.

European customers can see potential trouble: Coroot intervenes deeply with the data flow of the systems it is intended to monitor, with no alternative to doing things any other way. After all, Coroot's central task is to identify threats by the traffic flowing through, which means that the data automatically leaves the European Union's General Data Protection Regulation (EU GDPR) area. Apart from the possibility that organizations might be opposed to their data being directed in this way, the compliance issue between the US Clarifying Lawful Overseas Use of Data (CLOUD) Act and the GDPR also plays a role. If this situation is not going to cause a compliance problem for you, Coroot is a very interesting tool.

Coroot is primarily aimed at operators of state-of-the-art systems that rely on container orchestration, which makes Kubernetes [5] a permanent fixture in this universe, and for good reason. Practically the entire Coroot model for tapping data relies on Kubernetes under the hood. What's more, Coroot is designed to collect the corresponding user data from containers only, which does not rule out anything; today, more or less any application can be containerized with relative ease. That said, administrators will want at least to take a closer look at the deployment model if they are considering Coroot.

The first step is to familiarize yourself with the solution's key components. Coroot itself acts as the platform's linchpin, on the one hand acting as an API and, on the other, handling communication with the provider's cloud services (i.e., the part that delivers the AI functions) in the background. Coroot relies on two services to maintain and manage its data, storing the metrics data in the well-known Prometheus time series database. Once again Prometheus proves to be the go-to tool when it comes to metrics data storage. Coroot stores logs, traces, and recognized profiles in ClickHouse [6].

ClickHouse is not a proprietary Coroot development but a column-based database that focuses on analyzing incoming data. A SQL dialect lets you automatically analyze the stored data on the basis of various parameters. If you are thinking of a kind of database for business intelligence, you are on the right track. Prometheus and ClickHouse can also be used as data sources, besides storing data, and are therefore a genuine bidirectional exchange.

From Coroot's point of view, the question of how to store data is at least as important as how to acquire the data to be analyzed. The developers have expanded their solution to use telemetry data in line with the OpenTelemetry standard [7] as the basis for its analyses. This approach primarily means two types of data: logfiles and traces. This data can be linked to existing applications in two ways: either with the OpenTelemetry SDK [8], which ensures deep integration at an application's API level, or by teaming the application with OpenTelemetry Collector, which writes logs and traces.

The World of eBPF

Coroot also can read metrics data, logs, traces, and existing monitoring profiles from Kubernetes clusters through their APIs. The Coroot cluster agent takes care of this process, but the unique selling point in terms of tapping data is the Coroot node agent, which relies on the extended Berkeley Packet Filter (eBPF) technology [9]. This technology has already been the subject of several articles [10]. Considering its power, it is surprising that eBPF is not more popular or at least more widely known.

eBPF is a component of the Linux kernel and enables certain programs to run in a type of virtual machine (VM) inside the kernel with full access to the network stack. Just as all packets of a system pass through Netfilter and can be manipulated with the help of nftables, they also flow through the in-kernel VMs that you configure with the aid of eBPF. All traffic flowing through can be analyzed at will and manipulated if needed.

As a comparatively modern technology, eBPF does not have any legacy ballast, which makes it as fast as lightning. Proof-of-concept projects are already on the market to implement a complete packet filter in eBPF. This setup is far faster than the traditional Netfilter, so it stands to reason that eBPF would attract the attention of a solution such as Coroot.

If Coroot is viewed as a monitoring tool in the broader sense, it faces practically the same challenges as any other such system. For something to be monitored, runtime data from a tool or application somehow needs to find its way to the monitoring tool.

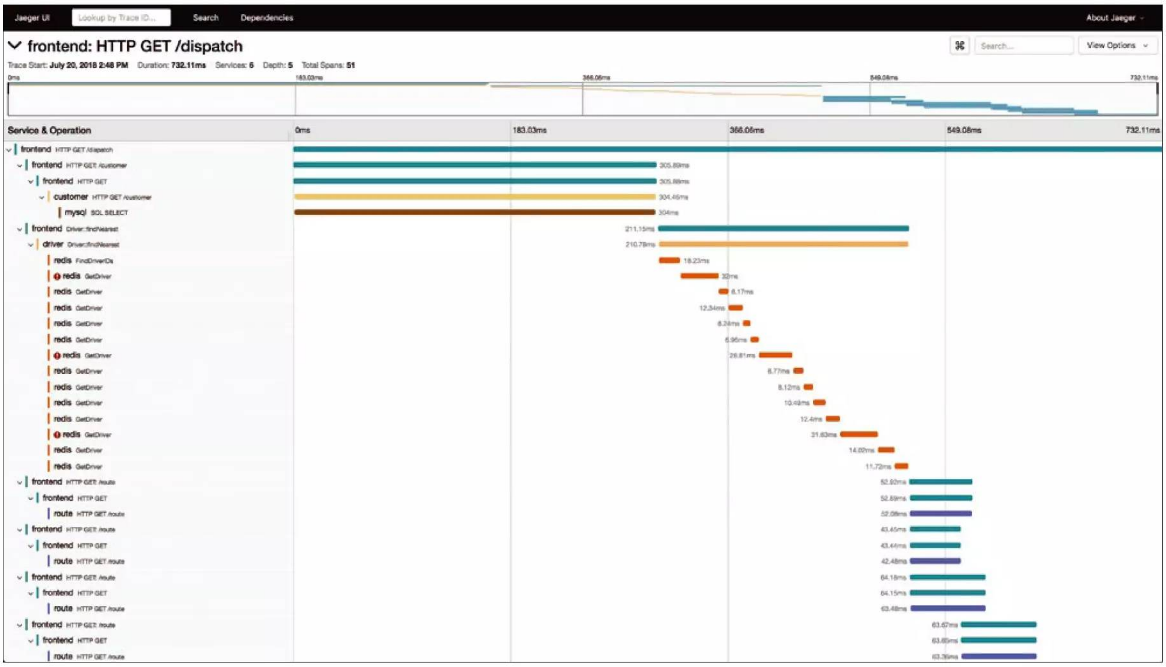

Jaeger UI
Lookup by Trace ID...
Search
Dependencies
About Jaeger
frontend: HTTP GET /dispatch
Trace Start: July 20, 2018 2:48 PM Duration: 732.11ms Services: 6 Depth: 5 Total Spans: 51
Service & Operation
0ms
183.03ms
366.06ms
549.08ms
732.11ms

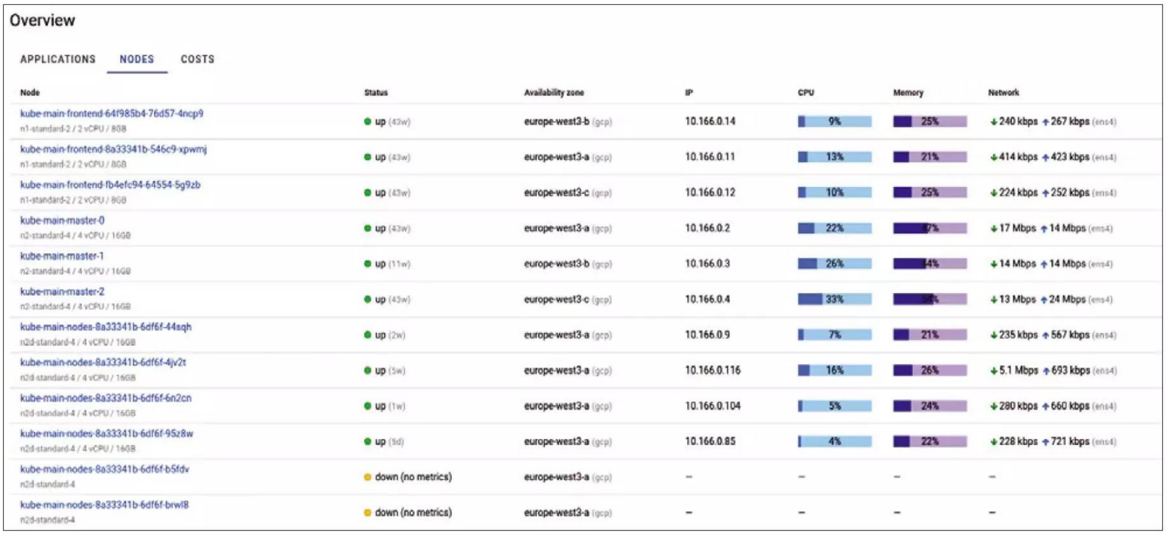
Figure 1: Administrators of distributed applications often use Jaeger to implement observability, so the application must actively output data to the observability tool because tracing would be impossible otherwise. © Jaeger

In techspeak, this is referred to as instrumentation (Figure 1).

If you look at legacy monitoring tools, the checks that admins store for certain services on individual systems do exactly this: They collect data as the object of monitoring. The problem in distributed environments, which automatically also means container environments, is their very dynamic nature, and legacy monitoring tools find it difficult to cope with this complexity. Moreover, metrics data is generated in so many different places that it would be a painstaking process to roll out a data collector for each and then integrate the data into the monitoring setup.

Today's microcomponent-based applications, in particular, add a whole new level of complexity. Besides monitoring the applications themselves, the individual pieces of information change in many ways on their way through the components of such a construct. Documenting these changes for monitoring purposes is often very useful or even essential.

In practice, the entire OpenTelemetry project spends most of its time


Overview

APPLICATIONS NODES COSTS

Node	Status	Availability zone	IP	CPU	Memory	Network
kube-main-frontend-64f985b4-76d57-4ncp9 n1-standard-2 / 2 vCPU / 8GB	● up (43w)	europe-west3-b (gcp)	10.166.0.14	9%	25%	↓240 kbps ↑267 kbps (ens4)
kube-main-frontend-8a33341b-546c9-xpwmj n1-standard-2 / 2 vCPU / 8GB	● up (43w)	europe-west3-a (gcp)	10.166.0.11	13%	21%	↓414 kbps ↑423 kbps (ens4)
kube-main-frontend-fb4efc94-64554-5g9zb n1-standard-2 / 2 vCPU / 8GB	● up (43w)	europe-west3-c (gcp)	10.166.0.12	10%	25%	↓224 kbps ↑252 kbps (ens4)
kube-main-master-0 n2-standard-4 / 4 vCPU / 16GB	● up (43w)	europe-west3-a (gcp)	10.166.0.2	22%	[illegible]	↓17 Mbps ↑14 Mbps (ens4)
kube-main-master-1 n2-standard-4 / 4 vCPU / 16GB	● up (11w)	europe-west3-b (gcp)	10.166.0.3	26%	[illegible]	↓14 Mbps ↑14 Mbps (ens4)
kube-main-master-2 n2-standard-4 / 4 vCPU / 16GB	● up (43w)	europe-west3-c (gcp)	10.166.0.4	33%	[illegible]	↓13 Mbps ↑24 Mbps (ens4)
kube-main-nodes-8a33341b-6df6f-44sqh n2d-standard-4 / 4 vCPU / 16GB	● up (2w)	europe-west3-a (gcp)	10.166.0.9	7%	21%	↓235 kbps ↑567 kbps (ens4)
kube-main-nodes-8a33341b-6df6f-4jv2t n2d-standard-4 / 4 vCPU / 16GB	● up (5w)	europe-west3-a (gcp)	10.166.0.116	16%	26%	↓5.1 Mbps ↑693 kbps (ens4)
kube-main-nodes-8a33341b-6df6f-6n2cn n2d-standard-4 / 4 vCPU / 16GB	● up (1w)	europe-west3-a (gcp)	10.166.0.104	5%	24%	↓280 kbps ↑660 kbps (ens4)
kube-main-nodes-8a33341b-6df6f-95z8w n2d-standard-4 / 4 vCPU / 16GB	● up (5d)	europe-west3-a (gcp)	10.166.0.85	4%	22%	↓228 kbps ↑721 kbps (ens4)
kube-main-nodes-8a33341b-6df6f-b5fdv n2d-standard-4	● down (no metrics)	europe-west3-a (gcp)	–	–	–	–
kube-main-nodes-8a33341b-6df6f-brwl8 n2d-standard-4	● down (no metrics)	europe-west3-a (gcp)	–	–	–	–

Figure 2: The Coroot user interface provides initial insights shortly after the solution goes live. It automatically recognizes a large proportion of the rolled-out resources thanks to its zero-instrumentation approach. © Coroot

The GNOME™ Conference

GUADEC

Brescia, Italy
July 24–29, 2025

GUADEC is GNOME's main annual event. Held since 2000, it brings together Free Software enthusiasts and professionals from all over the world.

Join us at GUADEC 2025, in Brescia or online, to hear about the latest technical developments, attend talks, participate in workshops, and celebrate GNOME!

Learn More

GUADEC.ORG

The GNOME Conference
GUADEC
2024
Denver, Colorado, USA
July 19–24

creating interfaces to which external tools can dock and that support an exchange of metrics data and tracing. I've already looked at Jaeger [11] in detail in this context; the tool evaluates OpenTelemetry-compatible traces, making it possible to track how data changes en route through a microcomponent application.

For this principle to work, either the application must be integrated with the OpenTelemetry SDK or the OpenTelemetry Collector must be able to access the data from outside to the greatest extent possible. In both cases, you need to add specific support for the required monitoring system to the app.

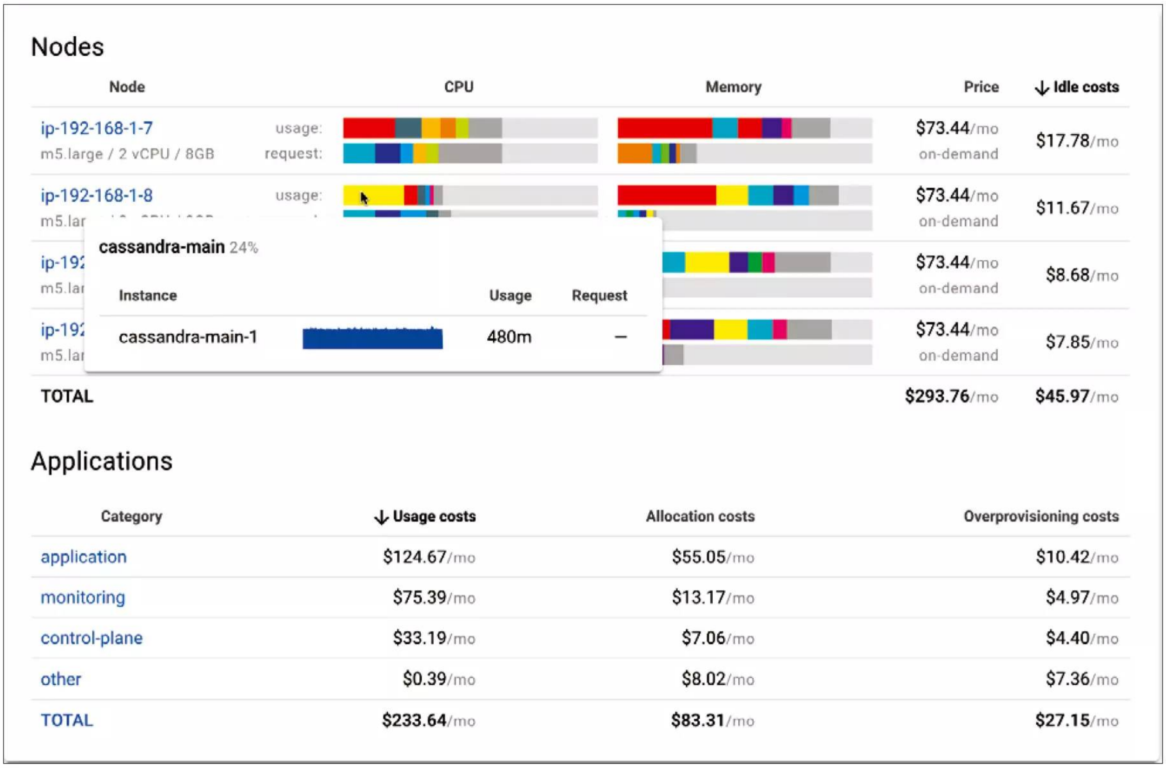
If you transfer the principle to a solution like Coroot, every application on any Kubernetes cluster anywhere in the world would then have to know about the existence of Coroot and actively feed it metrics, logs, and traces, which is not convenient or trivial. Therefore, the Coroot developers took a different approach with eBPF to extract and process a large part of the data required for analysis directly from an application's network traffic. This zero-instrumentation approach is attractive from an admin perspective. Once Coroot integration has been rolled out for a cluster, the data for monitoring automatically migrates from there to Coroot for analysis. Unlike other scenarios, you no longer need to connect each individual app, which is true – another important point – irrespective of how the application changes during its runtime.

For example, if Kubernetes starts more instances of the same service with a scale set to balance higher loads, the use of `coroot-node-agent` automatically takes it into account. The component hooks into eBPF on each system belonging to the cluster to intercept the relevant data. This arrangement sounds good in theory, and it works extremely well in practice: Coroot starts fielding comprehensive metrics, logs, and traces shortly after commissioning in a cluster, which means it can start analyzing the data straightaway (Figure 2).

Opportunities

The Coroot developers have greatly expanded the scope of their solution in recent years. Whereas the main selling point was initially that of detecting threat scenarios with AI and writing comprehensive, automated root cause analyses (RCAs), the focus is now more broadly the entire topic of observability. Although AI-supported attack detection still plays a major role, Coroot is now capable of doing significantly more. A few examples quickly illustrate this facility.

The Coroot developers have expanded their product to include the option of evaluating accounts with hyperscalers in terms of cost and efficiency. This challenge is familiar: Amazon AWS, Microsoft Azure, and Google Cloud Platform (GCP) make it child's play to provision and use virtual infrastructure, but none of the three providers view themselves as IT charities. If you don't pay close attention to the costs per hour during deployment, you could be in for a nasty surprise



Nodes

Node		CPU	Memory	Price	↓ Idle costs
ip-192-168-1-7 m5.large / 2 vCPU / 8GB	usage: request:			$73.44/mo on-demand	$17.78/mo
ip-192-168-1-8 m5.lar	usage:			$73.44/mo on-demand	$11.67/mo
ip-192 m5.lar				$73.44/mo on-demand	$8.68/mo
ip-192 m5.lar				$73.44/mo on-demand	$7.85/mo
TOTAL				$293.76/mo	$45.97/mo

cassandra-main 24%

Instance		Usage	Request
cassandra-main-1		480m	–

Applications

Category	↓ Usage costs	Allocation costs	Overprovisioning costs
application	$124.67/mo	$55.05/mo	$10.42/mo
monitoring	$75.39/mo	$13.17/mo	$4.97/mo
control-plane	$33.19/mo	$7.06/mo	$4.40/mo
other	$0.39/mo	$8.02/mo	$7.36/mo
TOTAL	$233.64/mo	$83.31/mo	$27.15/mo

Figure 3: In Kubernetes setups for hyperscalers, Coroot automatically correlates resource usage and computes the incurred charges. It also points out ways of optimizing the setup to help you save money. © Coroot

at the end of the month. If you supply the login credentials for a cloud account, Coroot will automatically connect to the API, read out the applicable prices, and generate an overview of the costs incurred.

Once again, Coroot uses AI to create a kind of dynamic analysis of the rolled-out setup and makes suggestions with a view to potential optimization. The mantra of the Coroot developers is clear: You need a comprehensive treasure trove of data at your disposal to be able to carry out comprehensive analyses, which is precisely where Coroot can help you. That said, Coroot's cost-tracking functions primarily focus on the hyperscalers' managed Kubernetes offerings. If you use Amazon Elastic (EKS), Microsoft Azure Kubernetes Service (AKS), or Google Kubernetes Engine (GKE), you can use Coroot to export and analyze metrics data automatically from rolled-out clusters (Figure 3).

If you automatically collect a wide range of data in your setup in the scope of threat detection, you can also use the data for other purposes. For example, the Coroot developers point out that the data pool is ideal for monitoring service level targets right down to individual instances of a single service whose availability can be permanently tracked by Coroot. The Coroot graphical user interface, which can draw comprehensible graphics from the available data, is a massive help in this scenario.

If you really want to, you can even use Coroot to record and plot individual outliers (e.g., in the response time of a service). Helpfully, Coroot can create a complete performance profile for practically any application in a cluster through just a few mouse clicks. The results show which individual functions were called and detail their processing times.

Observability is not something you implement just for fun; however, Coroot makes many things easy. Where you previously had to spend hours rolling out companion services and entire observability platforms, Coroot is almost fun. A number of correlations that were previously completely hidden from your eyes can now be displayed thanks to Coroot teamed with eBPF.

If you want to trace the communication paths of the applications in a virtual Kubernetes setup comprising several containers, for example, you often used to have to sit down for hours with a pen and paper. Thanks to Coroot, a map displaying all the communication flows in a setup is virtually a side product of the monitoring that takes place anyway (Figure 4).

Comprehensive Alerting

The practical use of an observability solution would be severely impaired if it could not inform anyone about impending trouble or about its findings in general. The Coroot developers offer a comprehensive solution by seamlessly connecting to various pager and operations services out of the box, including PagerDuty [12] and Opsgenie [13]. Alerting can also be set up in this way.

If you prefer a more down-to-earth approach, you can connect Coroot directly to Slack or Microsoft Teams and receive messages by instant messenger in the traditional way. A webhook option is also available that involves Coroot simply calling up a URL when a defined event occurs;

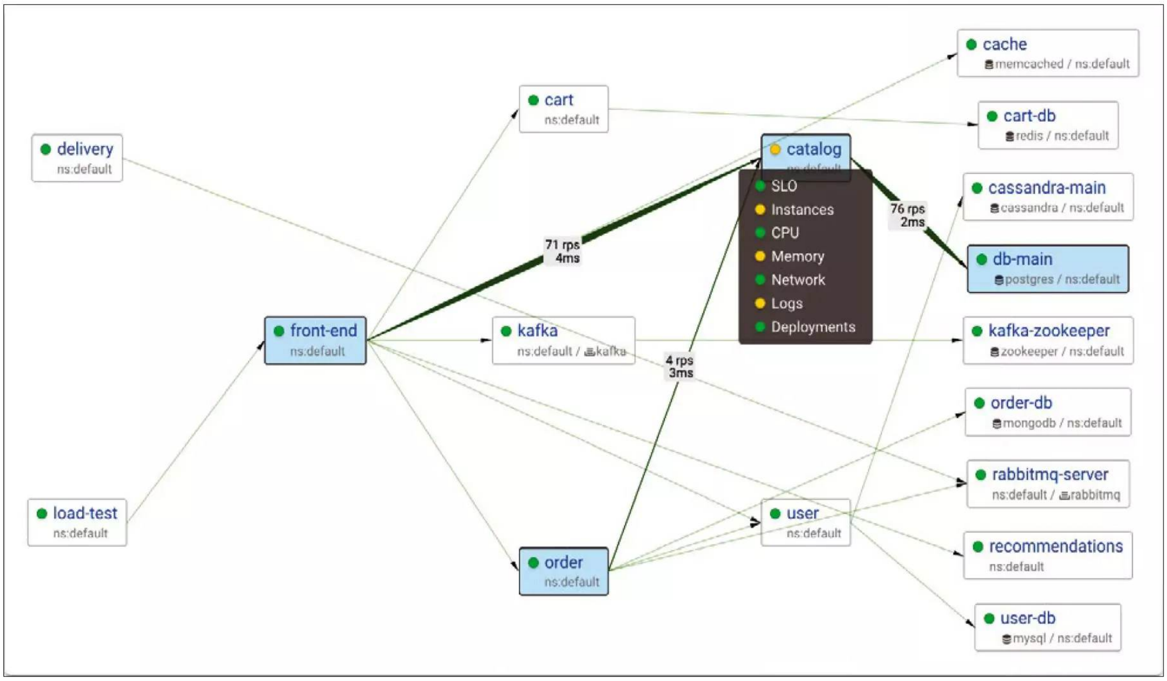

cache
memcached / ns:default
cart
ns:default
cart-db
redis / ns:default
delivery
ns:default
catalog
ns:default
SLO
Instances
CPU
Memory
Network
Logs
Deployments
76 rps
2ms
cassandra-main
cassandra / ns:default
71 rps
4ms
db-main
postgres / ns:default
front-end
ns:default
kafka
ns:default / kafka
kafka-zookeeper
zookeeper / ns:default
4 rps
3ms
order-db
mongodb / ns:default
rabbitmq-server
ns:default / rabbitmq
load-test
ns:default
user
ns:default
order
ns:default
recommendations
ns:default
user-db
mysql / ns:default

Figure 4: Coroot generates a complete matrix of communication solely from recorded data that would otherwise involve a huge amount of manual work. © Coroot

the URL can trigger practically any action, including sending email or text messages.

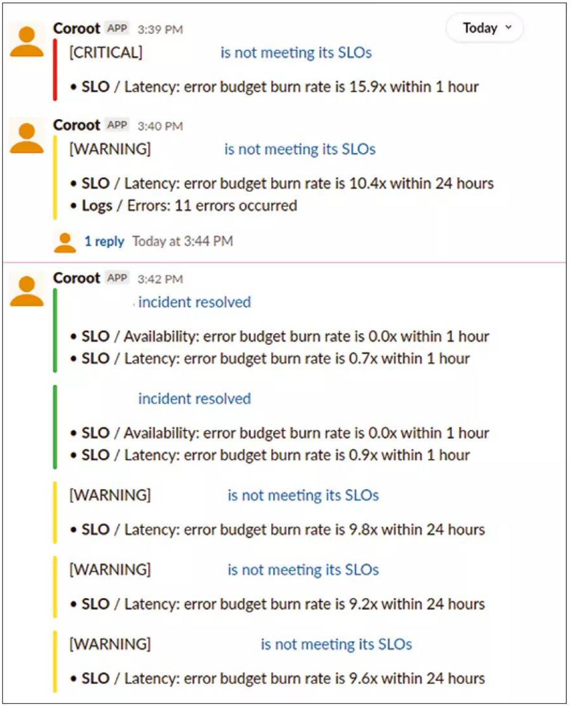
All the functions described so far can be linked to alerting. If Coroot is used to monitor service level objectives (SLOs), it can immediately alert you if certain threshold values are exceeded, and you can then take a look at the situation.

Prices

Although Coroot itself is open source, many tracing functions, especially those based on AI functions, can only be accessed in the cloud, for which you need to create an account with Coroot; the Coroot instance in your data center then uses this account for communication.

Coroot's pricing model is not likely to cause any heart attacks. Standard and premium editions differ from each other primarily in terms of available support. The premium edition also offers proactive monitoring according to the rate, error, duration (RED) principle, which means raising an alarm if certain threshold values are exceeded in the scope of SLO monitoring (Figure 5).

This feature is not available in the Standard Edition, which costs $1 per month and monitored CPU core, although Coroot offers volume discounts. The vendor does not publicly list the prices for the Premium Edition – quite possibly they are subject to individual agreements with respective customers.



Coroot APP 3:39 PM
[CRITICAL] is not meeting its SLOs
- SLO / Latency: error budget burn rate is 15.9x within 1 hour

Coroot APP 3:40 PM
[WARNING] is not meeting its SLOs
- SLO / Latency: error budget burn rate is 10.4x within 24 hours
- Logs / Errors: 11 errors occurred

1 reply Today at 3:44 PM

Coroot APP 3:42 PM
incident resolved
- SLO / Availability: error budget burn rate is 0.0x within 1 hour
- SLO / Latency: error budget burn rate is 0.7x within 1 hour

incident resolved
- SLO / Availability: error budget burn rate is 0.0x within 1 hour
- SLO / Latency: error budget burn rate is 0.9x within 1 hour

[WARNING] is not meeting its SLOs
- SLO / Latency: error budget burn rate is 9.8x within 24 hours

[WARNING] is not meeting its SLOs
- SLO / Latency: error budget burn rate is 9.2x within 24 hours

[WARNING] is not meeting its SLOs
- SLO / Latency: error budget burn rate is 9.6x within 24 hours

Figure 5: **If you promise your customers certain service targets, you need to be able to monitor compliance. Coroot can map SLOs and automatically check for compliance.** © Coroot

Conclusions

In some ways, Coroot feels like a glimpse into the future of observability in distributed setups. Where slow solutions such as Nagios and the like previously handled monitoring, today's technical requirements have exceeded their capabilities and brought new approaches such as Coroot to the fore.

The tool stands out in several respects. It doesn't need extensive instrumentation that developers have to wrap around their apps for things to work; instead, it relies on eBPF to collect the relevant data. Coroot is not only capable of analysis by profiling individual components, but it can also evaluate metrics data from Prometheus and analyze logfiles. It uses all of these sources to provide in-depth insights into the current status of a container-based setup. Additional features such as automated threat detection, comprehensive analysis of the quality of a service provided, and comprehensive cost monitoring for cloud-based services round things out.

Unlike other solutions, organizations do not need to worry about making provisions for insolvency if they want to use Coroot, because the pricing is reasonable. That said, European admins might find that the need to access a non-European cloud for a large part of the Coroot functionality spoils the fun. ■

Info

[1] Coroot: [https://coroot.com]

[2] SpamAssassin: [https://spamassassin.apache.org]

[3] Rspamd: [https://rspamd.com]

[4] "Four Solutions for Prometheus Long-Term Storage" by Martin Gerhard Loschwitz, *ADMIN*, issue 85, 2025, pg. 24, [https://www.admin-magazine.com/Archive/2025/85/Four-solutions-for-Prometheus-long-term-storage]

[5] Kubernetes: [https://kubernetes.io]

[6] ClickHouse: [https://clickhouse.com]

[7] OpenTelemetry: [https://opentelemetry.io]

[8] OpenTelemetry SDK: [https://opentelemetry.io/docs/languages/sdk-configuration/general/]

[9] eBPF: [https://ebpf.io]

[10] "Get Deeper Insights into Your System with eBPF" by Mayank Sharma, *Linux Magazine*, issue 225, August 2019, pg. 30, [https://www.linuxpromagazine.com/Issues/2019/225/Getting-Insights-with-eBPF/]

[11] "Tracking down problems with Jaeger" by Martin Gerhard Loschwitz, *ADMIN*, issue 70, 2022, pg. 56, [https://www.admin-magazine.com/Archive/2022/70/Tracking-down-problems-with-Jaeger/]

[12] PagerDuty: [https://www.pagerduty.com]

[13] Opsgenie: [https://www.atlassian.com/software/opsgenie]

The Author

Martin Loschwitz is the founder and managing director of True West IT Services GmbH, which offers scalable IT infrastructure based on OpenStack and Kubernetes.

Hone Your Skills
– with –
Special Issues!

JOIN THE LINUX REVOLUTION!
GETTING STARTED WITH LINUX
FIND THE PERFECT TOOL FOR THE TASK
Cool Linux Hacks
Tricks and shortcuts for Linux geeks
114 HACKS
301 BEST BASH COMMANDS
LINUX SHELL HANDBOOK
SUPERCHARGE YOUR LINUX SKILLS
LibreOffice Expert
Dive deep into the world's greatest free office suite
THE BEST OF LINUX MAGAZINE 2024
Advanced Techniques from Linux Experts:

Get to know Shell, LibreOffice, Linux, and more from our Special Issues library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

RASPBERRY PI GEEK THE COMPLETE ARCHIVE
MakerSpace
HANDS-ON PROJECTS FOR MAKERS
BUILD A RASP PI RADIO!
MakerSpace #02
RASP PI TRICKS FEED YOUR FISH WHILE YOU'RE AWAY
MakerSpace #03
RASP PI TRICKS TRY ARTIFICIAL INTELLIGENCE ON A PI
MakerSpace #04
BUILD INFO DISPLAYS
MakerSpace #05
Go Retro with DOS & BASIC
10+ Creative Projects

background image © roystudio, 123RF.com

Check out the full library!

shop.linuxnewmedia.com/shop

Develop a simple scanning application with the NAPS2 SDK

Digital Docs

A NAPS2 PDF scanner, the C# language, and the .NET Framework set the stage for a CLI app for scanning documents. By Andrea Ciarrocchi

A simple command-line application that scans one or more documents and saves them as images (JPEG, PNG, TIFF) and PDFs uses an SDK provided by the Not Another PDF Scanner 2 (NAPS2) developers, the C# language, and the .NET framework. NAPS2 is a free, open source, document-scanning application that allows users to scan, organize, and save documents [1].

NAPS2 supports various scanners and offers optical character recognition (OCR), batch scanning, and easy document editing, all with a simple, user-friendly interface. Although the NAPS2 application and SDK are cross-platform, the drivers and libraries related to the system's graphical interface vary depending on the operating system. Therefore, the application in this article will be specific to Linux. However, you can adapt the code to a different environment by making a few simple modifications.

Setting Up the Environment

I recommend setting up an integrated development environment like Visual Studio Code (VSCode) [2] for C# and .NET programming and having a basic understanding of the language and the framework. On Ubuntu, an easy way to install VSCode is by downloading the `.deb` package, right-clicking the file, and selecting the *Install* option.

Alternatively, you can search for *vscode* in the App Center and click the *Install* button. If you prefer the terminal, you can find detailed instructions for any Linux distribution online [3]. I also suggest adding C# and .NET development extensions, including *.NET Install Tool*, *C#*, *C# Dev Kit*, and *Code Runner* to manage and run the project.

The .NET framework is necessary, and it is officially supported on various distributions. On Ubuntu, the easiest way is to proceed through the App Center. The .NET website [4] provides more detailed information about the installation and configuration of the framework.

Creating the Project Solution

Once the VSCode extensions are installed, you create a new project by selecting *File* | *New File* | *.NET New Project* (Figure 1). Alternatively, you could proceed in the terminal with the `dotnet new` command; however,

Lead Image © elnur, 123RF.com

VSCode and its extensions greatly simplify project management at every stage.

The project folder will contain subfolders named `.vscode`, `bin`, and `obj`, as well as files with the extensions `.csproj`, `.cs`, and `.sln`. The `.csproj` file contains the list of installed NuGet packages and other project properties, such as the version of the .NET library being used. A NuGet package is an archive that contains compiled code, metadata, and dependencies for reusable .NET libraries or tools; it simplifies adding, updating, and managing dependencies in .NET projects.

The `.cs` file contains the application code, and the `.sln` file serves as a container for multiple related projects and configurations, specifying the relationships between projects, their build configurations (e.g., *Debug* or *Release*), and target platforms (e.g., *Any CPU* or *x64*). The file also stores global solution settings, including the build order and dependencies and may include logical groupings of projects or files for better organization.

To add NuGet packages, just press Ctrl + Shift + P and enter the name of the desired package (Figure 2). A list of entries matching the search appears. Simply select the correct package from the list and the most recent available version. The packages needed for this project are *NAPS2.Images*, *NAPS2.Images.Gtk*, and *NAPS2.Sdk*. After adding them, the `.csproj` file should look similar to Listing 1.

Developing the Application

The SDK provides good encapsulation of scanning functions, and the resulting code is quite straightforward (Listing 2). This code demonstrates how you can use the NAPS2 library to build a simple application for scanning documents and exporting them as a PDF.

To start, you import the necessary namespaces from the library (e.g., `NAPS2.Images`, `NAPS2.Scan`, and `NAPS2.Pdf`), which provide all the functionality needed to manage images, control the scanner, and handle PDF exports. For the GTK-based

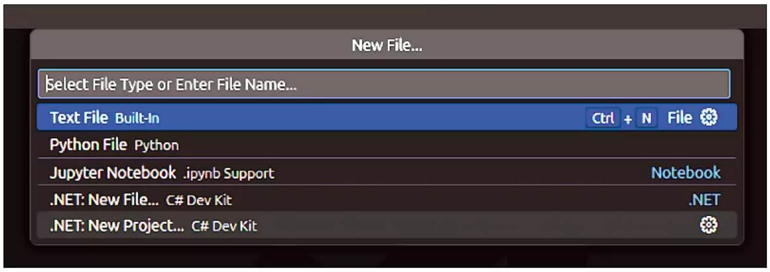

New File...
Select File Type or Enter File Name...
Text File Built-In
Ctrl + N File
Python File Python
Jupyter Notebook .ipynb Support
Notebook
.NET: New File... C# Dev Kit
.NET
.NET: New Project... C# Dev Kit

Figure 1: Creating a new .NET project.

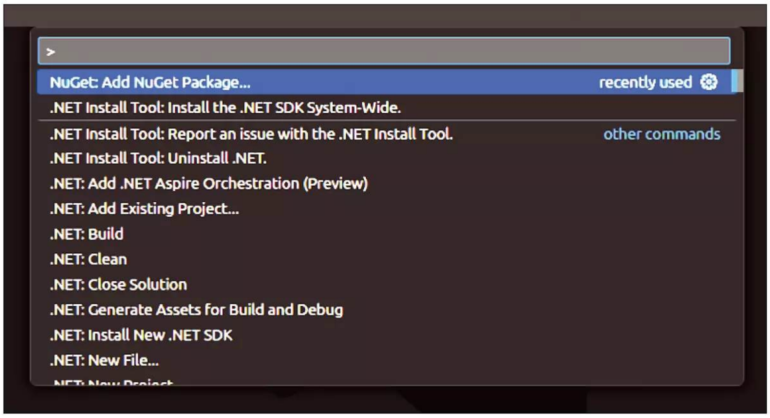

>
NuGet: Add NuGet Package...
recently used
.NET Install Tool: Install the .NET SDK System-Wide.
.NET Install Tool: Report an issue with the .NET Install Tool.
other commands
.NET Install Tool: Uninstall .NET.
.NET: Add .NET Aspire Orchestration (Preview)
.NET: Add Existing Project...
.NET: Build
.NET: Clean
.NET: Close Solution
.NET: Generate Assets for Build and Debug
.NET: Install New .NET SDK
.NET: New File...

Figure 2: Adding a NuGet package.

Listing 1: `myScan.csproj`

```
<PackageReference Include="NAPS2.Images" Version="1.1.3" />
<PackageReference Include="NAPS2.Images.Gtk"
    Version="1.1.3" />
<PackageReference Include="NAPS2.Sdk" Version="1.1.3" />
```

Listing 2: `Program.cs`

```
using NAPS2.Images;
using NAPS2.Images.Gtk;
using NAPS2.Scan;
using NAPS2.Pdf;

public class HelloWorldSample
{
  static async Task Main(string [] args)
  {
    await HelloWorldSample.Run();
  }

public static async Task Run()
{
  using var scanningContext = newScanningContext(new
     GtkImageContext());
  var controller = new ScanController(scanningContext);

  ScanDevice device = (await controller.
     GetDeviceList()).First();

  var options = new ScanOptions
  {
    Device = device,
      PaperSource = PaperSource.Feeder,
      PageSize = PageSize.A4,
      Dpi = 300
  };

  var images = new List<NAPS2.Images.ProcessedImage>();

  int i=1;
  Console.WriteLine("Scan in process..");
  await foreach (var image incontroller.Scan(options))
  {
    image.Save($"page{i++}.jpg");
      images.Add(image);
  }
  Console.WriteLine("Scanning complete!");

  var pdfExporter = new PdfExporter(scanningContext);
  await pdfExporter.Export("doc.pdf",images);
  }
}
```

image-handling context, also include *NAPS2.Images.Gtk*. To ensure the application works on Windows, you should use the *NAPS2.Images.Gdi* library instead.

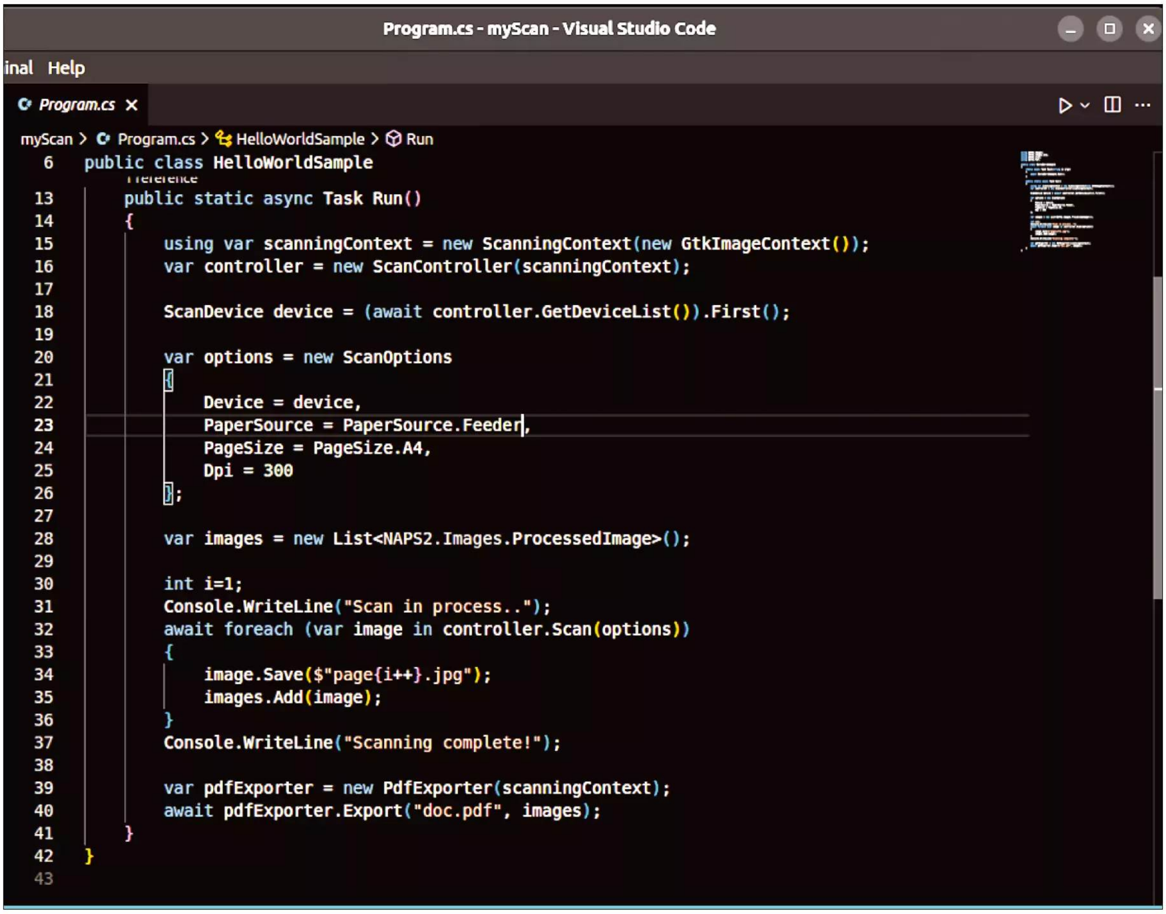
The program begins with the definition of the `HelloWorldSample` class, which contains the `Main` method entry point, where you invoke the `Run` method asynchronously. This design ensures that the program can handle time-consuming operations like scanning and file export without blocking the main thread.

In the `Run` method, you first establish a scanning environment by creating a `ScanningContext` object, which is configured to use a `GtkImageContext` that provides the necessary tools to process scanned images in a GTK-based graphical environment. Next, you initialize a `ScanController` object in this scanning context, which acts as the main interface for interacting with the scanner.

Next, retrieve the list of available scanning devices by calling the `GetDeviceList` method on the controller. From this list you select the first device to proceed with the scan. With the device selected, the `ScanOptions` object sets up the scanning options. Here, you specify that the feeder tray will be the paper source, the page size will be A4, and the scan resolution will be 300DPI. The scanner parameters are highly customizable. For instance, to change the document source from the feeder to the scanner's flatbed, you simply need to set the `PaperSource` option to `Flatbed`. You can find complete documentation for the available APIs online **[5]**.

Once the scanning options are configured, you begin the scanning process by creating an empty list to store the scanned images and an `await foreach` loop to process each image as it is produced by the scanner. As each page is scanned, you save it as a JPEG file, naming the files sequentially (e.g., `page1.jpg`, `page2.jpg`, etc.). Also, each processed image is added to the list, ensuring you have all the scanned pages ready for the final PDF export.

During the scanning process, you print messages to the console to indicate progress, letting the user know that the scan is underway and when it is complete. This basic feedback provides a better user experience. After all the pages have been scanned, you move on to the final step: creating a PDF by initializing a `PdfExporter` object in the same scanning context used earlier. The `Export` method of the `PdfExporter` combines all the scanned images into a single PDF file named `doc.pdf`, completing the process. The program ends after successfully generating the PDF.


Program.cs - myScan - Visual Studio Code
inal Help
Program.cs
myScan > Program.cs > HelloWorldSample > Run

```
public class HelloWorldSample
    public static async Task Run()
    {
        using var scanningContext = new ScanningContext(new GtkImageContext());
        var controller = new ScanController(scanningContext);

        ScanDevice device = (await controller.GetDeviceList()).First();

        var options = new ScanOptions
        {
            Device = device,
            PaperSource = PaperSource.Feeder,
            PageSize = PageSize.A4,
            Dpi = 300
        };

        var images = new List<NAPS2.Images.ProcessedImage>();

        int i=1;
        Console.WriteLine("Scan in process..");
        await foreach (var image in controller.Scan(options))
        {
            image.Save($"page{i++}.jpg");
            images.Add(image);
        }
        Console.WriteLine("Scanning complete!");

        var pdfExporter = new PdfExporter(scanningContext);
        await pdfExporter.Export("doc.pdf", images);
    }
}
```

Figure 3: Portion of the VSCode.

The primary drawback of the NAPS2 SDK is its limited adoption, which makes it challenging to find support information when needed.

Running the Application

To start the application [6], click the *Run Code* button (typically a Play icon in the top right corner; Figure 3). Alternatively, you can run the project with the command

```
dotnet run
```

after ensuring you are in the folder where the `.csproj` file is located. Assuming you are scanning a three-page document, the output will be similar to that shown in Figure 4. At the end of the process, you will find three JPEG files and `doc.pdf` in the project folder, representing the conversion of the scanned pages into a single PDF.

Conclusion

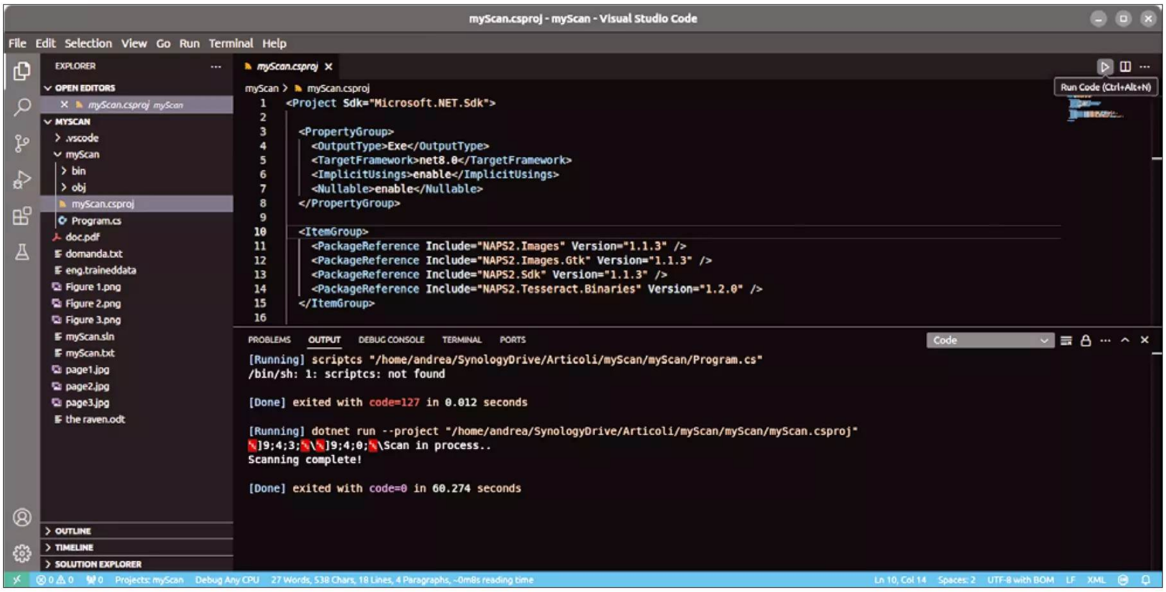
Although not a production sample, this code demonstrates how to build a document-scanning and PDF export utility. It efficiently integrates scanning, image processing, and file export into a seamless workflow, providing a foundation on which you can expand for more complex applications, such as adding OCR or scanner sharing. ■

Info

[1] NAPS2: [https://naps2.com/]
[2] VSCode: [https://code.visualstudio.com/download]
[3] VSCode terminal install: [https://code.visualstudio.com/docs/setup/linux]
[4] Install .NET on Linux: [https://learn.microsoft.com/en-us/dotnet/core/install/linux]
[5] NAPS2.Sdk: [https://www.naps2.com/sdk/doc/api]
[6] Code in this article: [https://linuxnewmedia.thegood.cloud/s/9nFQcFb2p8oRMEJ]

The Author

Andrea Ciarrocchi is a technology enthusiast. You can find out more about his work at [https://andreaciarrocchi.altervista.org].


myScan.csproj - myScan - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
myScan.csproj myScan
MYSCAN
.vscode
myScan
bin
obj
myScan.csproj
Program.cs
doc.pdf
domanda.txt
eng.traineddata
Figure 1.png
Figure 2.png
Figure 3.png
myScan.sln
myScan.txt
page1.jpg
page2.jpg
page3.jpg
the raven.odt
OUTLINE
TIMELINE
SOLUTION EXPLORER
myScan.csproj
myScan > myScan.csproj
Run Code (Ctrl+Alt+N)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Code
Projects: myScan
Debug Any CPU
27 Words, 538 Chars, 18 Lines, 4 Paragraphs, ~0m8s reading time
Ln 10, Col 14
Spaces: 2
UTF-8 with BOM
LF
XML

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="NAPS2.Images" Version="1.1.3" />
    <PackageReference Include="NAPS2.Images.Gtk" Version="1.1.3" />
    <PackageReference Include="NAPS2.Sdk" Version="1.1.3" />
    <PackageReference Include="NAPS2.Tesseract.Binaries" Version="1.2.0" />
  </ItemGroup>

```

```
[Running] scriptcs "/home/andrea/SynologyDrive/Articoli/myScan/myScan/Program.cs"
/bin/sh: 1: scriptcs: not found

[Done] exited with code=127 in 0.012 seconds

[Running] dotnet run --project "/home/andrea/SynologyDrive/Articoli/myScan/myScan/myScan.csproj"
␛]9;4;3;␛\␛]9;4;0;␛\Scan in process..
Scanning complete!

[Done] exited with code=0 in 60.274 seconds
```

Figure 4: List of imported NuGet packages.

Comparing system rescue distros

First Aid Kit

Rescue systems play a particularly important role for desktops because they cannot simply be reinstalled on the fly. We look at venerable but outdated Knoppix and three still in development: Grml, SystemRescue, and Finnix.

By Martin Gerhard Loschwitz

The heyday of rescue systems for Linux and other computers is over: Popular solutions such as Rescatux have been orphaned or their development has at least been paused for several years. Knoppix isn't doing too much at the moment either, although version 9.1 from January 2021 is still available [1] and was featured in the June 2021 *Linux Magazine* [2]. Developer Klaus Knopper is now very busy as a lecturer and Vice President for Digitalization at the Kaiserslautern University of Applied Sciences.

At least three genuine rescue systems still exist and are under active development: Grml, SystemRescue, and Finnix. In this article, I take a closer look at the trio and also at the veteran Knoppix.

Knoppix

Anyone who has spent a few years in IT is guaranteed to have some kind of memories of Knoppix (Figure 1), which was considered for many years the best choice for trying out the free Linux operating system. The system was largely developed and maintained by Klaus Knopper and enjoyed massive fame because of its inventor, especially within the German-speaking Linux community. Knopper has long been a respected keynote speaker at conferences and trade fairs such as LinuxTag. What's more, in

Figure 1: Knoppix is regarded as the pioneer of Live Linux CDs and developed into a rescue system later, but its development has now been largely discontinued.

Lead Image © Kirill Makarov, 123RF.com

the distribution's heyday, numerous administrators used Knoppix to repair systems (e.g., damaged in the course of updates). In this respect, Knoppix has always been regarded as a powerful rescue system.

In the meantime, however, the former buzz surrounding Knoppix has died down. The last Knoppix version, dated 2022, was an exclusive supplement to the German *Linux-Magazin*; in fact, it is not even available for download from the official Knoppix website.

However, another reason explains why the relevance of Knoppix and, more fundamentally, rescue systems has dwindled. Where data center administrators used to turn to Knoppix to help a system get back on its feet, lifecycle management solutions and automation now rule the day.

The push toward containerization by the major vendors, such as Red Hat and Canonical, has also meant that systems now require significantly less support. Distribution upgrades from one major version of a system to the next used to be a nerve-wracking task. Today, upgrades mainly involve updating central operating system tools, with the applications updated as containers. Systemd does the rest to ensure that classic sources of error, such as changes to the network configuration of systems, no longer automatically mean a disaster. Therefore, in data centers in particular, the need for rescue systems that can be carried around on a USB storage medium has dropped massively in recent years.

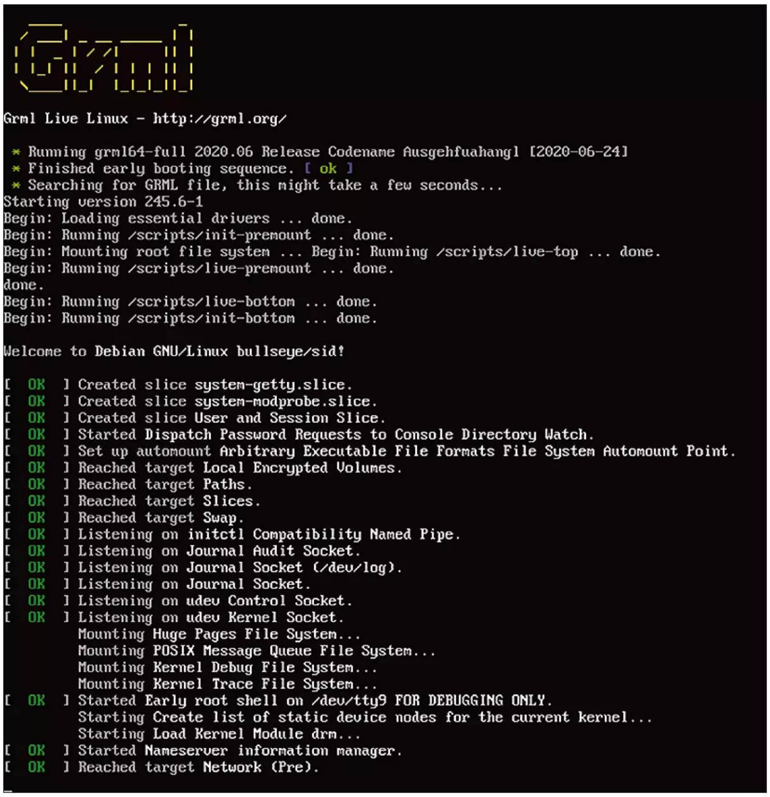
Of course, rescue systems are still needed, especially on desktops, but the rules of the game are different. Users don't want to delete their everyday work device completely in the event of problems, set it up again, and then restore the valuable data afterward with a backup. Even on desktops, serious system failures are far rarer today than they were in the past because components such as the Linux kernel and parts of the basic system are now far more robust than before and because the developers of typical desktop components now also integrate rollback options into their tools in the event of a failure.

Another reason for the decline of rescue systems is that every popular desktop distribution comes with a Live image of itself that allows you to boot and modify an existing system. Every Ubuntu CD, for example, can be booted to a Live desktop, as can current images of openSUSE or Fedora Core. That said, the risk of seeing an error message instead of the usual splash screen after restarting your Linux system has by no means been ousted from desktops today and is far greater there than on servers, simply because desktops usually see more frequent updates, which implicitly increases the risk of something breaking. Therefore, people who use Linux on the desktop still need a system like Knoppix, but the only available version is outdated, so the question of a suitable alternative arises.

If you search the web for up-to-date Linux rescue systems, you always come across the same candidates: Grml, SystemRescue (previously known as SystemRescueCd), and Finnix are the three remaining major options.

Grml

Grml [3] is something of a veteran among rescue systems (Figure 2). The distribution has been around since 2005, and its lead maintainer is still Austrian Michael Prokop, who is also a Debian developer. Little wonder, then, that the Grml substructure consists of Debian GNU/Linux, which was not always the case. In the early days, Grml was based on Knoppix, and only later did Prokop change the substructure. Grml development has also slowed down considerably; the current Grml version dates back to February 2024, which means it is still up to date and can be used on current systems, particularly because it uses kernel 6.6. Incidentally, the system gets its name from the onomatopoeic



```
Grml Live Linux - http://grml.org/

 * Running grml64-full 2020.06 Release Codename Ausgehfuahangl [2020-06-24]
 * Finished early booting sequence. [ ok ]
 * Searching for GRML file, this might take a few seconds...
Starting version 245.6-1
Begin: Loading essential drivers ... done.
Begin: Running /scripts/init-premount ... done.
Begin: Mounting root file system ... Begin: Running /scripts/live-top ... done.
Begin: Running /scripts/live-premount ... done.
done.
Begin: Running /scripts/live-bottom ... done.
Begin: Running /scripts/init-bottom ... done.

Welcome to Debian GNU/Linux bullseye/sid!

[  OK  ] Created slice system-getty.slice.
[  OK  ] Created slice system-modprobe.slice.
[  OK  ] Created slice User and Session Slice.
[  OK  ] Started Dispatch Password Requests to Console Directory Watch.
[  OK  ] Set up automount Arbitrary Executable File Formats File System Automount Point.
[  OK  ] Reached target Local Encrypted Volumes.
[  OK  ] Reached target Paths.
[  OK  ] Reached target Slices.
[  OK  ] Reached target Swap.
[  OK  ] Listening on initctl Compatibility Named Pipe.
[  OK  ] Listening on Journal Audit Socket.
[  OK  ] Listening on Journal Socket (/dev/log).
[  OK  ] Listening on Journal Socket.
[  OK  ] Listening on udev Control Socket.
[  OK  ] Listening on udev Kernel Socket.
         Mounting Huge Pages File System...
         Mounting POSIX Message Queue File System...
         Mounting Kernel Debug File System...
         Mounting Kernel Trace File System...
[  OK  ] Started Early root shell on /dev/tty9 FOR DEBUGGING ONLY.
         Starting Create list of static device nodes for the current kernel...
         Starting Load Kernel Module drm...
[  OK  ] Started Nameserver information manager.
[  OK  ] Reached target Network (Pre).
_
```

Figure 2: Grml is a genuine Linux rescue system that offers a GUI but does not force you to use it. Command-line mode is just as easy to use.

© Grml/Michael Prokop

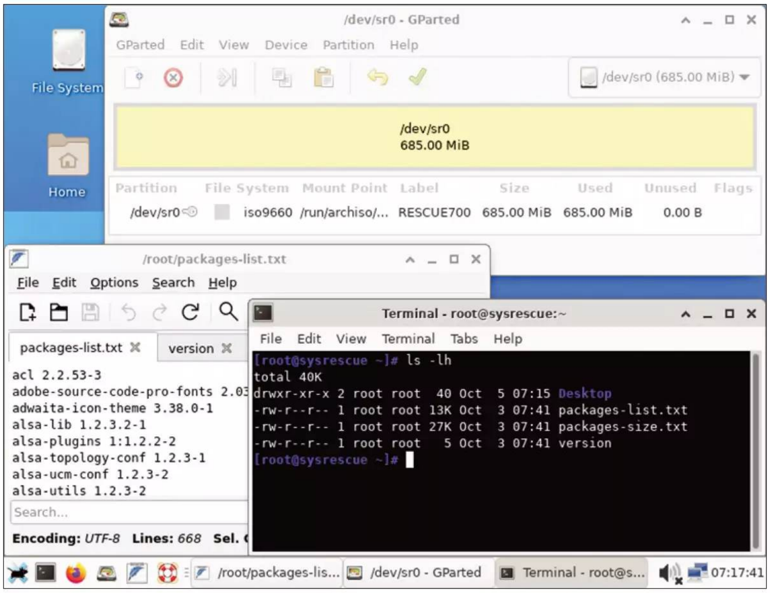
description of the noise that administrators make when they set off to rescue a system with Grml, for which the distribution is perfectly equipped. Although desktop environments such as KDE and Gnome are missing, Grml comes with the far leaner alternatives Openbox and Fluxbox. Although you can use a graphical user interface (GUI) in Grml, it will typically be used as a terminal multiplexer (i.e., to display many terminals simultaneously).

The GUI is not mandatory. If you prefer to work with the plain vanilla console, you can easily do so in Grml. The many configuration and repair tools included in Grml will be of most interest to users. When it comes to filesystem repair, for example, Grml comes with a full kit of tools that help you inspect and repair all popular Linux filesystems, as well as the logical volume manager (LVM). Software RAID configurations are also easy to inspect and restore with Grml if a repair is at all technically possible.

In particular, the modes offered by Grml for its operation have improved in recent years. Originally, the choice of software offered by Grml was primarily driven by the desire to save space, which applied to both the data carrier on which Grml would be written and to the volume of data that Grml would load onto running systems in the absence of local disk storage. The original idea was that Grml should fit on a normal CD and work without complaint on older systems low on RAM.

However, both restrictions have now been removed. Servers or even desktops with a CD drive are becoming increasingly rare, and many admins no longer have a CD burner to create the matching media. Also, even a fairly spartan laptop rarely comes with less than 8GB of RAM today, which easily is enough to run even a larger Grml image. Nevertheless, Michael Prokop is not wasteful with RAM on the target systems; in fact, Grml still has a very small footprint. You no longer have to worry about CD-ROMs either: `grml2usb` has been around for years, so it is possible to write an executable Grml installation to a standard USB storage device in next to no time. If you want to do yourself a favor, grab a USB storage device with USB 3.0, which has long been standard.

As icing on the cake, Grml has an option to install a complete Debian system on a target system with `debootstrap`. This component is the same ultimately used by the Debian installer itself. All told, Grml is targeted more at server admins, but it also allows experienced desktop users to restore a system after encountering an issue. Essential tools such as `chroot` are included. Grml is up to date and comes with a modern kernel, as well, which makes it far easier to identify contemporary hardware.



Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
/dev/sr0	iso9660	/run/archiso/...	RESCUE700	685.00 MiB	685.00 MiB	0.00 B	

```
acl 2.2.53-3
adobe-source-code-pro-fonts 2.03
adwaita-icon-theme 3.38.0-1
alsa-lib 1.2.3.2-1
alsa-plugins 1:1.2.2-2
alsa-topology-conf 1.2.3-1
alsa-ucm-conf 1.2.3-2
alsa-utils 1.2.3-2
```

Encoding: *UTF-8* **Lines:** *668* **Sel.**

```
[root@sysrescue ~]# ls -lh
total 40K
drwxr-xr-x 2 root root  40 Oct  5 07:15 Desktop
-rw-r--r-- 1 root root 13K Oct  3 07:41 packages-list.txt
-rw-r--r-- 1 root root 27K Oct  3 07:41 packages-size.txt
-rw-r--r-- 1 root root   5 Oct  3 07:41 version
[root@sysrescue ~]# 
```

Figure 3: SystemRescue was previously designed for use on CDs and comes with an Xfce GUI. The distribution offers all kinds of practical rescue tools such as `ddrescue`.

© SystemRescue

SystemRescue

SystemRescue [4] is also a real veteran in the rescue distribution business (Figure 3). The previous name, SystemRescueCd, used up to 2020, indicated the type of target medium for which the distribution was initially designed: Like Grml, SystemRescue was intended to be stored on a CD that you would take to the data center to carry out emergency work. However, CDs would fall short today because the average image size of a SystemRescue installation is now at least 800MB. Instead, the system now comes as an ISO image that can be written to an USB storage device, and like Grml, the SystemRescue makers offer their own tool to do so.

Another parallel between SystemRescue and Grml is a GUI: SystemRescue uses the more convenient Xfce. The current version of the distribution is 11.02 dating from August 29, 2024; it also has Linux version 6.6.47 on board, which should be recent enough for any typical hardware setup. Under the hood, SystemRescue is based on Arch Linux with the Pacman package manager. SystemRescue also offers a permanent installation mode, but in this case, you end up with SystemRescue on your hard drive instead of a genuine Debian in Grml's case that naturally also works well without Grml.

SystemRescue comes with a versatile selection of tools. Desktop software for quick research is included in the form of Firefox 128.1.0, although Chromium is missing.

“Small but powerful” is the motto, but SystemRescue also has a few highlights, undoubtedly including a complete toolchain for all popular Linux filesystems, as well as tools for repairing multiple device (MD) RAID drives or LVM volumes. Vim is included in the current version 9.1, making it easy to edit configuration files. Common programming languages such as Python and Perl are also on board, making SystemRescue a useful platform for rolling out your own repair tools.

A Samba service can be run in SystemRescue, and like Grml, GNU Parted and the matching GUI GParted are present for partitioning disks. Backup tools such as `ddrescue` can read at least parts of a block device, even if other parts are defective. The highlight is the NTFS-3G driver, which can be used to patch Windows systems, if needed, clearly showing that SystemRescue does not only target Linux users, although Windows users who have never experienced a Linux command line might be a little out of their depth. SystemRescue is not an accessible general-purpose system but is clearly aimed at experienced sys admins with a Linux background who know what they are doing. If it is not possible to rescue a Windows system on the fly, SystemRescue can at least back up the system data so that it can subsequently be restored on another Windows computer.

Common debugging tools such as `ping` and `traceroute` round out the offering.

SystemRescue is therefore aimed not only at admins who want to repair a specific system but also addresses those situations in which you need a working system with a working shell on a specific network. Because of this flexibility, many companies that use SystemRescue resort to a hack and make the system available as a standard option in a netboot procedure.

In this way, SystemRescue is available on every system that boots off the network, along with the Memtest memory tester, which the distribution also has on board.

Whether administrators choose Grml or SystemRescue is primarily a question of personal preference. If you have a history with Debian or Ubuntu, you will intuitively find Grml easier to use, but if you are more at home in the Arch world, you will find SystemRescue a convincing alternative.

Finnix Linux

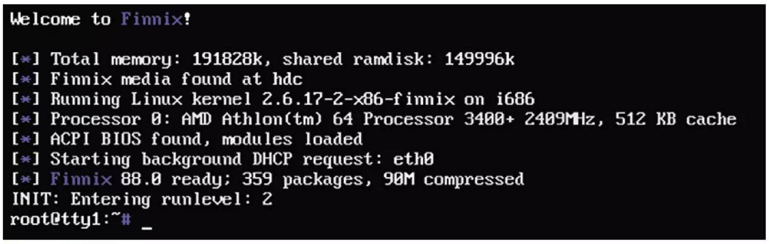
Finnix Linux [5] is something of a paradox in itself (Figure 4). The distribution dates back to 1999, and yet even many die-hard Linux admins have never heard of it. Anyone thinking that Finnix comes from Finland is also wrong. Instead, the product is named after its main developer Ryan Finnie, whose core motivation, as for Knoppix, Grml, SystemRescue, and various other distributions, was to create a rescue system for broken Linux installations.

However, it is up to the observer to decide to what extent this history of origins is coherent, because Finnix was initially based on Red Hat Linux 6.0. Knoppix later served as the basis of the system, until Ryan Finnie finally made the switch to Debian GNU/Linux, primarily using packages from the testing branch of the Debian distribution since then. Finnix has never forgotten its origins: The principle that Finnix has to be as lean as possible still applies today, with a basic Finnix system weighing in at just 400MB, making Finnix the only distribution in this comparison that would still easily fit on a normal CD. Of course, the Finnix developers also offer options for installing the system on a USB stick and running it from there.

When it comes to hardware, Finnix is equally frugal. Less than 1GB of RAM is required to use the system in a meaningful way, so that even ancient computers can be revived with Finnix. The developers themselves even state that the minimum requirement is only 32MiB but add the remark that Finnix could certainly use more RAM if available.

Of the test subjects in the comparison, Finnix is also the most up-to-date system. The last Finnix release was somewhat longer ago than that of SystemRescue, but because Finnix uses Debian’s testing branch, a current Linux kernel 6.8 is available, as well as a more modern toolchain for the basic system. Finnix also offers the best support in terms of target platforms. In contrast to SystemRescue, for example, Finnix supports 32-bit Intel CPUs without any restrictions, and Finnix can also be used on PowerPC systems, but only up to and including Finnix 111. Additionally, the Finnix developers provide the `finnix-live-build` tool that can also build Finnix ISO images for other architectures, with the ARM and ARM64 architectures playing a major role. Finnix for ARM64 can certainly be an attractive option for those who manage Mac systems with Apple’s M processor, because it can basically run on these chips. However, the Finnix people wisely refrain from making any promises with regard to any form of usability.

This purism in terms of distribution size or supported platform takes its toll on the practical side. Because Finnix lacks a GUI, on the one hand, it is easier to maintain the



```
Welcome to Finnix!

[*] Total memory: 191828k, shared ramdisk: 149996k
[*] Finnix media found at hdc
[*] Running Linux kernel 2.6.17-2-x86-finnix on i686
[*] Processor 0: AMD Athlon(tm) 64 Processor 3400+ 2409MHz, 512 KB cache
[*] ACPI BIOS found, modules loaded
[*] Starting background DHCP request: eth0
[*] Finnix 88.0 ready; 359 packages, 90M compressed
INIT: Entering runlevel: 2
root@tty1:~# _
```

Figure 4: Unlike Grml and SystemRescue, Finnix takes a purist approach by not including a GUI with the system. The subsequent additional work is rewarded with a lean 480MB footprint for the entire operating system image. © Finnix/Ryan Finnie

distribution because the developers can avoid problems with Xorg, Wayland, and other desktops. On the other hand, the approach also ensures that Finnix only targets experienced professional admins who know how to call up and use the terminals on their Finnix system.

In terms of the tools required for system recovery and restoration, however, Finnix does not leave any gaps. Most filesystems can be checked and repaired with Finnix, as can encrypted drives with `cryptsetup`, MDRAID drives under Linux, or LVM volumes. The distribution also includes a backup tool for copying data from a local disk to another location, should you ultimately want to reinstall a system. Of course, Finnix also includes tools for copying a finished Finnix image to a CD or USB storage device. Although Finnix would easily fit on a regular CD with 480MB, most admins today are more likely to use a USB storage device or make Finnix available as a system in a netboot environment.

Live Environments

Even if Knoppix is no longer maintained as regularly as it once was, various ideas from Klaus Knopper live on in the entire community, as the Live CD images that all modern desktop distributions have in their portfolio today bear witness. Knopper's idea of pressing an entire Linux system onto a CD and making it bootable was a major success. Although the idea of bootable Live systems had already been conceived by resourceful minds before Knopper, he made the idea so popular that it gradually spread to the entire Linux world.

Some veteran Linux users will remember the regular discussions about installing Knoppix on a hard disk because of its ease of use. Knoppix was not originally conceived as a plain vanilla rescue system, but was primarily intended to make Linux accessible without needing to install. When others saw that the idea worked, they followed suit. Canonical, for example, offered ISO images of Ubuntu at a relatively early stage that booted directly into a Live system from which the installation of the operating system could then be launched. If this approach reminds you of Grml, you are quite right – Grml is also one of the pioneers in this respect.

Today, the world of Live environments is quite confusing, primarily because every self-respecting distribution has one. For example, if you download the latest ISO images of Ubuntu 24.04, you can either burn them to a DVD or write them to a USB storage device and boot your computer directly from there. UnionFS and its various offshoots and successors make this possible: Linux, when booted from the storage device, behaves in the same way as would a permanently installed Linux (Figure 5). Another important factor here is that, strictly speaking, Live Linux today is no longer read-only and no longer needs to keep all of its content in RAM. Unlike the CDs and DVDs of the Knoppix years, USB storage devices are ultimately writable, which is where UnionFS comes into play. Administrators today

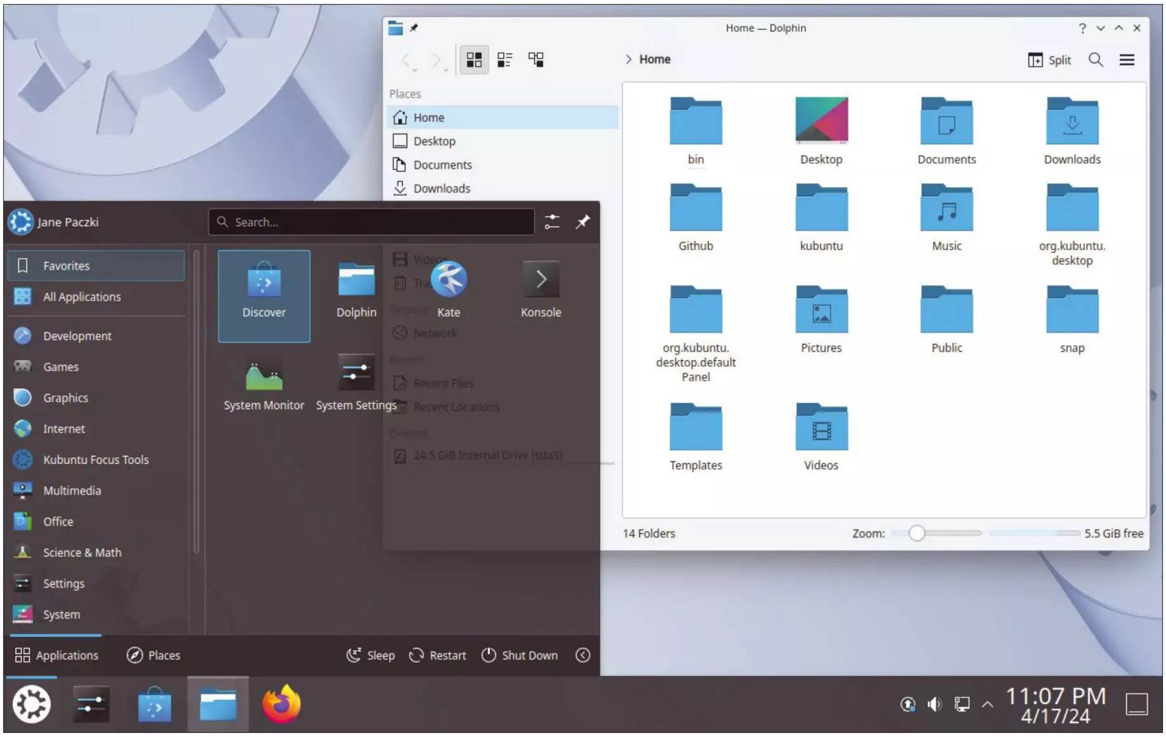

Home — Dolphin
Home
Split
Places
Home
Desktop
Documents
Downloads
bin
Desktop
Documents
Downloads
Github
kubuntu
Music
org.kubuntu.desktop
org.kubuntu.desktop.default Panel
Pictures
Public
snap
Templates
Videos
14 Folders
Zoom:
5.5 GiB free
Jane Paczki
Search...
Favorites
All Applications
Development
Games
Graphics
Internet
Kubuntu Focus Tools
Multimedia
Office
Science & Math
Settings
System
Discover
Dolphin
Kate
Konsole
System Monitor
System Settings
Applications
Places
Sleep
Restart
Shut Down
11:07 PM
4/17/24

Figure 5: The Live version of Kubuntu 24.04 and the installed version on the hard drive are not different in any major way. However, Kubuntu is not a specialist rescue system. © Canonical

regularly have a USB storage device at hand that contains a finely tuned rescue system and can be modified. As mentioned earlier, it was only the later Knoppix versions that focused more strongly on system tools that could also be used to make repairs. Unfortunately, the Live systems offered by today's crop of distributions are not particularly well prepared for disaster recovery. Take a look at the current Kubuntu or openSUSE Live images, for example, and you will have a good idea of the focus and functionality of the systems. Standard packages that are usually installed on the system anyway might also include various tools that make it possible to rescue a local system, and some distributors might even offer a Recovery menu item in the bootloader menu of the Live image to let you boot a different or differently configured system from your storage device. At the end of the day, however, correcting errors and rescuing existing systems is not the focus of the Live versions of desktop distributions. Tools such as `ddrescue` can be installed at a later date – provided you know what you are doing and what to look out for. From this point of view, it is therefore advisable to use one of the specialized distributions such as Grml, SystemRescue, or Finnix for genuine rescue missions.

Conclusions

If you take a look around DistroWatch today, you will quickly see that the heyday of rescue systems for Linux and other computers is over. Popular solutions such as Rescatux are now orphaned, or their development has at least been paused for several years. Not much is happening in the Knoppix camp either, but at least three genuine rescue systems still exist and are under active development: Grml, SystemRescue, and Finnix. Which of these solutions you choose is primarily a matter of personal preference. Any of the three cases won't be a bad choice.

Info

[1] Knoppix: [http://www.knopper.net/knoppix/index-en.html]

[2] "On the DVD," *Linux Magazine*, issue 247, June 2021, pg. 6, [https://www.linuxpromagazine.com/Issues/2021/247]

[3] Grml: [https://grml.org/]

[4] SystemRescue: [https://www.system-rescue.org/]

[5] Finnix: [https://www.finnix.org/]

The Author

Martin Loschwitz is the founder and managing director of True West IT Services GmbH, which offers scalable IT infrastructure based on OpenStack and Kubernetes.

IT Highlights at a Glance

Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox.

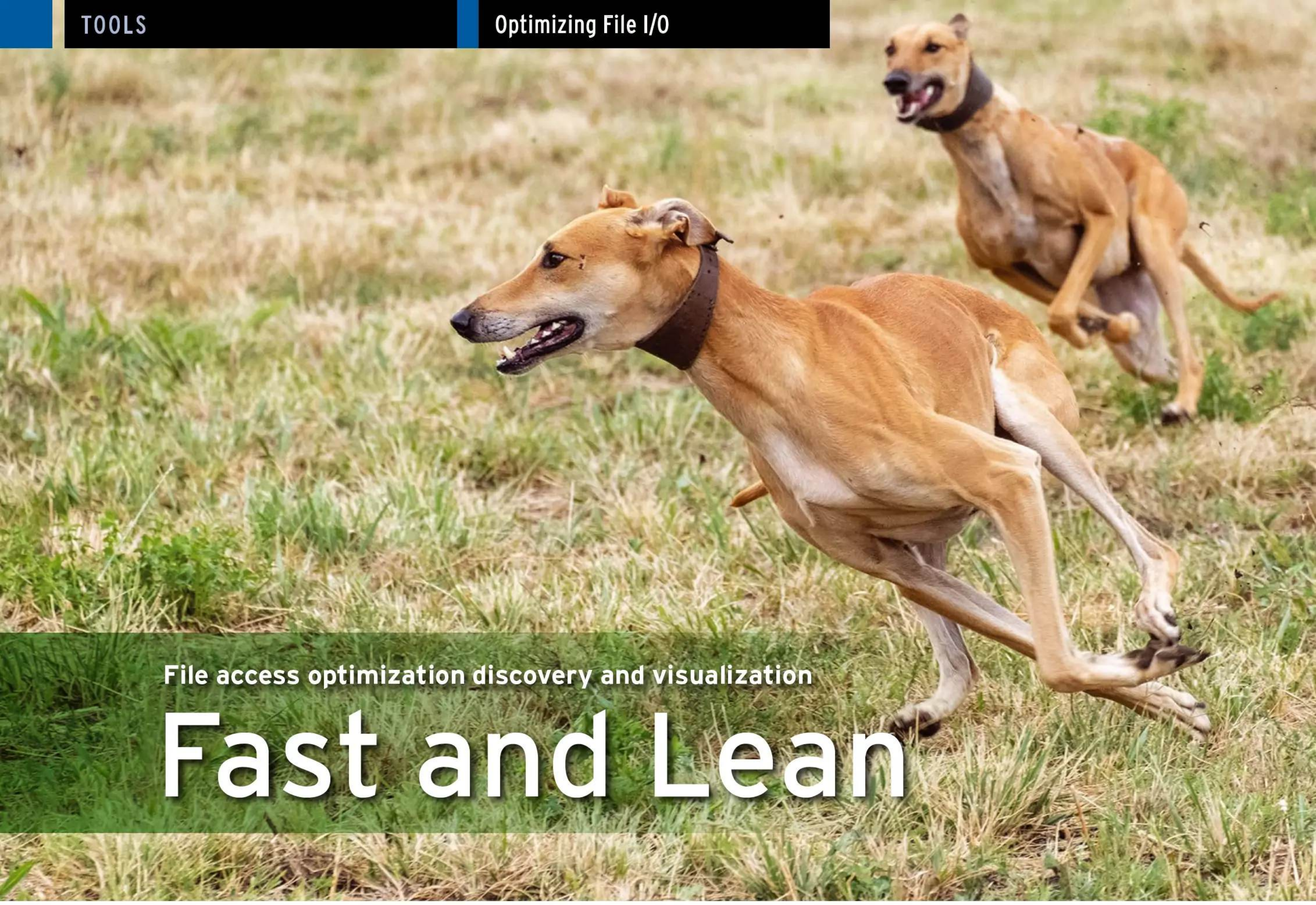
ADMIN Update • Linux Update

Keep your finger on the pulse of the IT industry.

ADMIN Update: bit.ly/ADMIN-Update
Linux Update: bit.ly/Linux-Update

Photo by Andrew Neel on Unsplash

ADMINUPDATE
LINUX UPDATE



File access optimization discovery and visualization

Fast and Lean

File I/O is a limiting factor for many applications and can become a bottleneck for data-intensive computing. We analyze different file access functions and present a script that visualizes and optimizes inefficient file I/O.

By Andreas Scheumann, Philipp Koester, and Rainer Keller

Analyzing program file I/O is difficult and time consuming, especially if you have to change an existing program with a long history of different developers. Therefore, assistance from tools is important.

Most existing tools (e.g., Darshan [1], Vampir [2], *libiotrace* [3]) are profilers that collect data during runtime and provide the collected data for later analysis. With the *libiotrace* profiling data, we developed an automatic analysis tool to find crucial patterns and to offer instructions to improve this pattern for POSIX [4] and MPI [5] file I/O (see the "Basics of File Access" box).

The underlying problem is inefficient file access that can cause a program to use more resources and time than necessary. One type of suboptimal file I/O is inefficient file access patterns. Our script [6] can help with the problem of unnecessary frequent and small reads and writes of data inside a file. In this article, we describe the methodology used for devising a script that serves as an example of the kinds of optimizations you can apply to your own homegrown applications to improve performance.

Basics of File Access

Before getting into the details, a good starting point is to describe how file access works internally and how several simultaneous file accesses on one file can interfere with each other.

Opening a file with low-level file access (e.g., in C) creates a file handle with a unique ID representing the opened file. In POSIX, the handles are called file descriptors or streams, whereas in MPI they are called MPI file handles. Because the underlying functionality is the same, the tool presented in this article addresses the different types of handles alike, and we generalize them by calling them active file accesses.

Both MPI and POSIX processes can open multiple files simultaneously, and an active file may have multiple concurrent accesses from a single process (Figure 1). The functions called on these active file accesses become meaningful, but do they interfere with each other, or can they, for example, be combined without changing the program behavior?

Optimization Possibilities

To achieve the aim of reducing the number of function calls per active file access, you must determine whether merging function calls is possible and consider parallel active file accesses on the same file. Therefore, three factors are important: the type of function used to access the file area, the time-based sequence of function calls onto a file, and the affected byte areas inside the file.

With this information, you can recognize whether the same function is called more than once successively and then check whether those calls can be merged into a single call. In the following section, we explain the theory of this check.

Photo by Barnabas Davoti on Unsplash

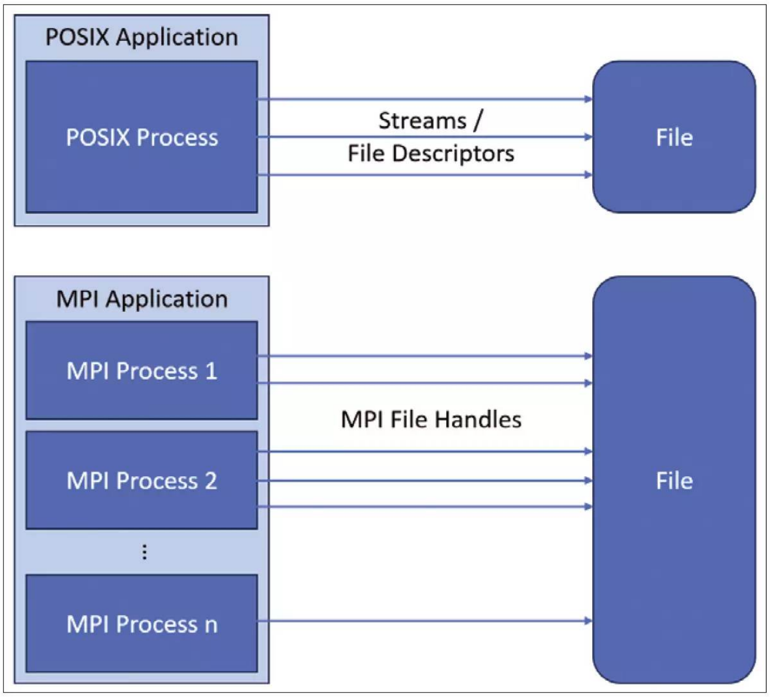


Figure 1: Illustration of POSIX and MPI application file handles.

Sequential Condition

When the script determines two function calls of the same type by a file access, the first step is to check for another function call by a different file access on the same file sequentially between the repeated function calls (hence called intermediate function). If no intermediate function call by a different file access occurs, you can merge calls into one, because the resulting file content will remain the same (Figure 2). If other file access function calls occur between the potentially optimizable calls, you need to investigate those cases further.

For simple cases (i.e., when the intermediate function between is only an open or close function), you can quickly rule out function interference.

The intermediate function in between neither changes the file nor depends on the repeated function calls. In this case, you can merge the repeated function calls and perform them in a single call before or after the intermediate function call.

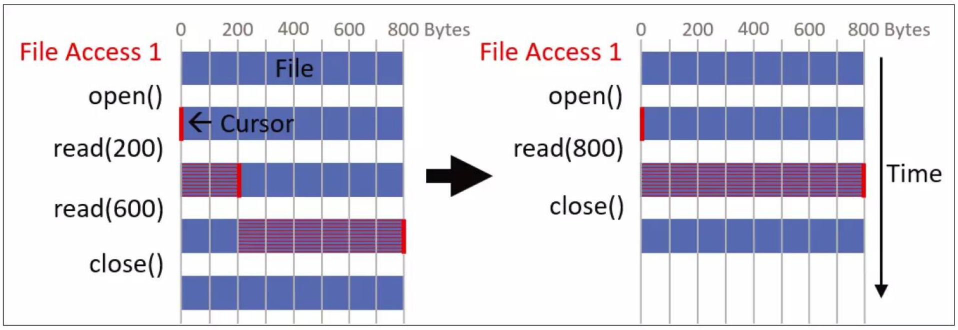


Figure 2: Merging two calls of the same type.

Conditions Dependent on the Offset

If the intermediate function instead is a seek, read, or write, you often need to consider additional factors. You can resolve some cases easily, but often the sequence of the function calls is not sufficient to make a statement about your ability to optimize. Often, the function calls and the corresponding byte areas within the file require a detailed look.

To get a better understanding of what is going on inside the file, it is important to know that each active file access has a so-called offset, which describes the current position of the cursor of the file access. Functions called by a process change the offset of the file handle according to the function type.

Functions Altering the Offset

The initial opening of a file sets the offset of the newly opened file handle at position zero. Calling a function that closes an active file handle removes the cursor from the file. Three types of functions change the offset according to a given parameter: seek, write, and read. The seek function jumps to a desired spot inside a file, changing the offset associated with the file handle, whereas write and read functions move the cursor position from its previous position.

Read functions read a specified number of bytes beginning from the current cursor position. After the function call, the offset increases by the number of bytes read. When you write bytes into a file, it inserts the content at the current cursor position. Like read functions, it moves the cursor to the end of the written block.

In this case, the new offset is the number of written bytes added to the previous offset. At the same time, a write function overwrites the content in its affected byte range or increases the file size when it reaches the end of the file.

Possible Cases

By keeping these offset-altering functions in mind, you can determine whether and how multiple active file accesses in the same file affect one another.

To structure all possibilities, we examine each potential repeated function type and analyze the different calls that can occur between them. When combining calls, we assume the application may allocate more memory for buffering file access.

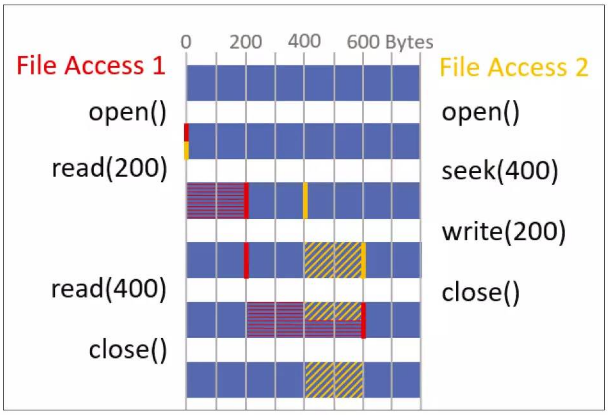

0
200
400
600 Bytes
File Access 1
open()
read(200)
read(400)
close()
File Access 2
open()
seek(400)
write(200)
close()

Figure 3: A write function call interfering with repeated read function calls.

Repeated open or close function calls will be excluded, because they do not typically happen. Multiple seek function calls that happen one after another are most possibly a coding issue, because a seek where no different function happens afterward has no use, and the first call is obsolete.

Repeated Reads

Read functions only read a sequence of bytes without modifying the file itself; therefore, in many cases, you can merge two sequential reads into one, which is always the case if the intermediate function calls do not change the file as well.

A seek call or another read call between the repeated function calls are cases in which you can easily merge the two read function calls and perform it before or after the function in between.

However, if a write call from another file access occurs on the file, you need to consider the byte area affected by the calls. Figure 3 shows a second read call of the first file access reading the bytes written by another file access that occurs after the first read call, which prevents simply merging the second call into the first because it would change the read result.

A write call that occurs within the read area between two read calls affects the content of the second read call, preventing the two reads from being merged. However, if bytes are written behind the read areas, the areas are non-overlapping, allowing a merge of two read calls and performing the new call before or after the write call (Figure 4).

Repeated Writes

When trying to merge two write calls interrupted by another function call by a different file access, you will find fewer cases in which you can merge the calls. The write calls alter the file, beginning from the position of the cursor at the time of the first function call until right before the cursor position after the second call. The function call between the write calls does not interfere if its byte area does not overlap the written areas (Figure 5). Any function not overlapping the written areas leads to mergeable write calls.

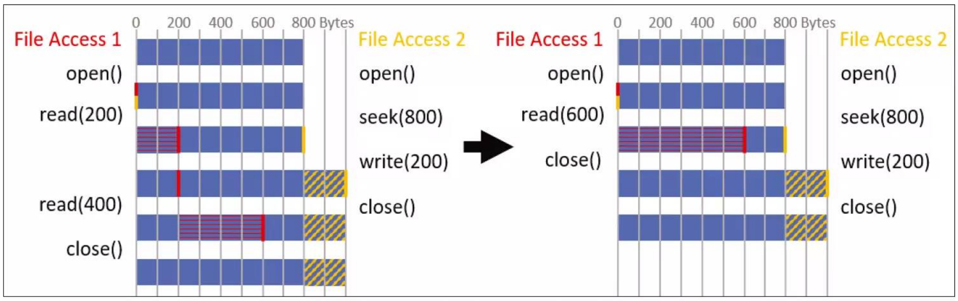

0
200
400
600
800 Bytes
File Access 1
open()
read(200)
read(400)
close()
File Access 2
open()
seek(800)
write(200)
close()
0
200
400
600
800 Bytes
File Access 1
open()
read(600)
close()
File Access 2
open()
seek(800)
write(200)
close()

Figure 4: Repeated read function calls with a non-interfering write in between can be optimizied.

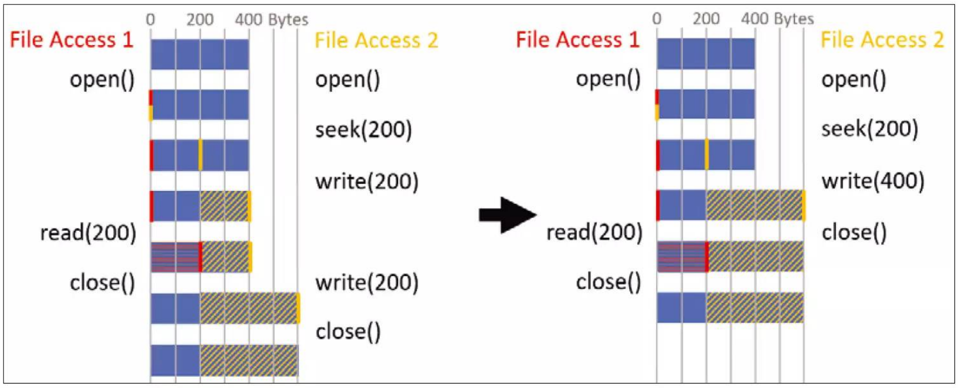

0
200
400 Bytes
File Access 1
open()
read(200)
close()
File Access 2
open()
seek(200)
write(200)
write(200)
close()
0
200
400 Bytes
File Access 1
open()
read(200)
close()
File Access 2
open()
seek(200)
write(400)
close()

Figure 5: Repeated write calls with a non-interfering read call in between.

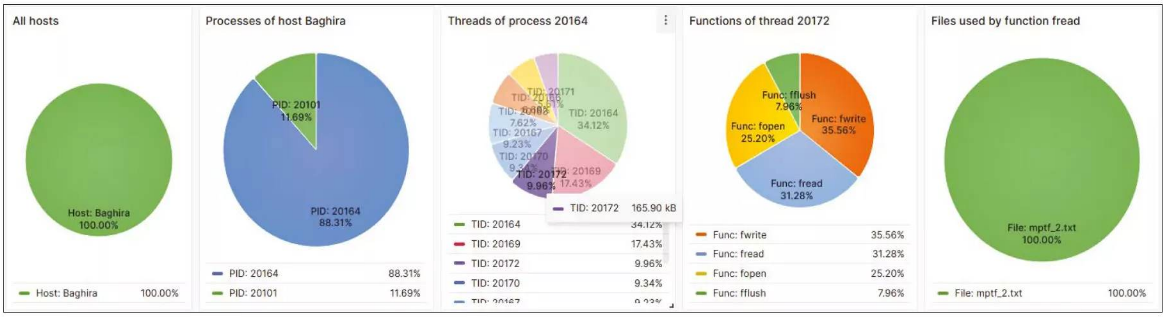

All hosts
Host: Baghira
100.00%
Host: Baghira 100.00%
Processes of host Baghira
PID: 20101
11.69%
PID: 20164
88.31%
PID: 20164 88.31%
PID: 20101 11.69%
Threads of process 20164
TID: 20171
TID: 20166
TID: 20168
7.62%
TID: 20164
34.12%
TID: 20167
9.23%
TID: 20170
9.34%
TID: 20172
9.96%
TID: 20169
17.43%
TID: 20172 165.90 kB
TID: 20164 34.12%
TID: 20169 17.43%
TID: 20172 9.96%
TID: 20170 9.34%
TID: 20167 9.23%
Functions of thread 20172
Func: fflush
7.96%
Func: fwrite
35.56%
Func: fopen
25.20%
Func: fread
31.28%
Func: fwrite 35.56%
Func: fread 31.28%
Func: fopen 25.20%
Func: fflush 7.96%
Files used by function fread
File: mptf_2.txt
100.00%
File: mptf_2.txt 100.00%

Figure 6: Pie chart panels of a Grafana dashboard showing the data from an InfluxDB time series database.

If any call happens inside the written areas, an optimization is impossible, because a merge would lead to inconsistencies.

Making It Better

The method discussed above forms the basis for determining whether repeated function calls can be merged to reduce their number and minimize the resulting overhead. Tracking programs and their file access patterns can be complicated and unintuitive while coding. Data about the file I/O of an application provides valuable insights into improving a program's efficiency. Tracing all function calls of a program with the necessary information and visualizing them helps you gain a deeper understanding, and the *libiotrace* [3] library can help.

libiotrace

This analysis tool monitors programs and collects file I/O profiling data, which sets a perfect entry point to analyze the program on the efficiency of its file I/O. The library traces file I/O-related function calls and provides additional useful information. It writes the data to an InfluxDB instance, storing each function call with the following details: the name of the function, its timestamp, the number of written or read bytes, the name of the file, and the related file handle.

Preparing the Data

As discussed, each active file access has a cursor in its opened file that describes its current position in the file. Libiotrace does not directly trace the current cursor position, also known as the offset, but you can find the function calls that alter the offset, along with additional information about the number of bytes the cursor is changed in the database. These details allow the determination of the current cursor offset before and after each function call.

A Python script [6] performs this task. With the *influxdb_client* Python library, the script loads time series *libiotrace* data from the InfluxDB instance. For easier handling, the script transforms the data into a `pandas.DataFrame` [7] and adjusts its structure.

In the next step, it processes the time-sorted data for each file access. Each function has different additional data traced by *libiotrace* that describes how the offset changes. With this information – and according to the function used – the script calculates the offset after each call and adds the value to a new column of the dataframe. The *influxdb_client* library writes the enriched data, in the form of a dataframe, to the InfluxDB instance for further use.

First Insights

A dashboard in Grafana enables an explorative visualization of the previously enriched InfluxDB data created by *libiotrace*. Both the tool's preparatory work and the finished solution involve the use of a Grafana dashboard to gain insights.

In the pre-work dashboard, interactive pie charts let you navigate from top to bottom, exploring the traced processes, function calls, and files used by the traced program. In the drill-down view shown in Figure 6, you can click on the pie charts to view more detailed information, progressing from left to right. This approach makes the *libiotrace* data accessible, allowing you to gain insights.

A connected pre-work time series diagram shows how the offset of file accesses on the selected file (Figure 7) and the cursor position of each file access changes over time. Each color stands for a distinct file access ID. Selecting a specific file access ID lets you see the cursor position's course

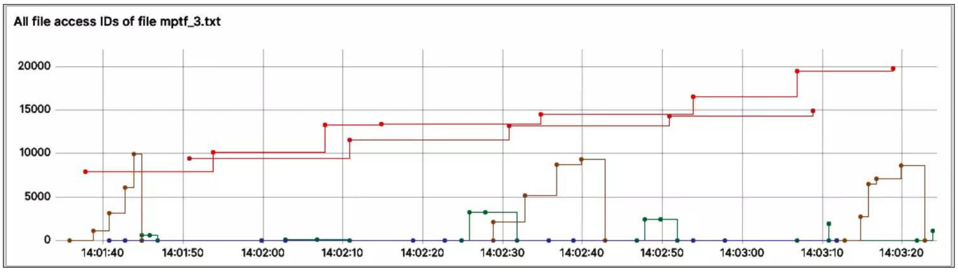


Figure 7: The course of the offsets of all file accesses on a selected file, distinguished by color.

in detail, with the called functions made visible (Figure 8). This method highlights when a function is repeated consecutively.

The ability to show function calls of other file accesses alongside the selected file access allows for a quick comparison of sequential calls. As you can see in Figure 9, the vertical lines display the timestamps of relevant functions of other file accesses on the same file and directly indicates when such a call occurs between two consecutive function calls of the same type by the selected file access.

An additional challenge arises during rapid function calls in a program. Libiotrace measures the timestamps with a very high level of detail to ensure consistency. Although InfluxDB can also handle this level of detail, Grafana only supports timestamps up to milliseconds in its dashboards.

For multiple points with timestamps differing only at the microsecond level or deeper, Grafana shows them as a single point in the time series diagrams, which results in missing data in the visualizations. More importantly, it prevents a visual sequential comparison.

The script tackles this issue because the sequence of the function calls is the most important property when looking for optimizations. When data points have different timestamps on a more detailed level, but the timestamps would be truncated to the same timestamp because of Grafana's granularity, the later timestamp is adjusted to the next millisecond.

Additionally, the script accounts for consecutive calls and adjusts their position when necessary to maintain the correct sequence. Grafana then recognizes both data points as different points and visualizes them as such. The script makes this small adjustment to the real timestamps to achieve the dashboard's goal of displaying the sequence of file accesses and their function calls.

This first step in identifying problematic situations where unnecessary frequent and small function calls occur has limitations, including the explorative nature of the dashboard, which makes the optimization process quite time consuming. Also, this visualization only covers the simple sequential comparison of file accesses and does not allow a detailed comparison at the offset level.

Instead of manually analyzing such cases with the dashboard, the goal was to automate the detection of these optimizations and visualize them in a clear and explanatory way. The tool explained in this article performs this automated optimization discovery.

Implementation Details

After the script processes the data created by *libiotrace* to include information about the current offset, the tool then uses this information to identify optimization opportunities. It does so by iterating over all file accesses and filtering the entire dataframe to include only the data of the current file access.

Each row in this dataframe contains information such as the timestamp, function name, offset after the call, and accessed file name, among other details. The script then iterates through the filtered dataframe row by row in chronological order, searching for consecutive occurrences of a function. A dictionary collects all rows involved in function calls that are being called more than once consecutively and bundles them in lists.

The next step iterates over the collected information inside the dictionary. For each function call bundle, the script takes the timestamps of the first and the last call. It uses these timestamps to filter the original dataframe of all file accesses on the current file. This retrieves all function calls from other file accesses that took place between the repeated function calls. This filtered dataframe contains all the

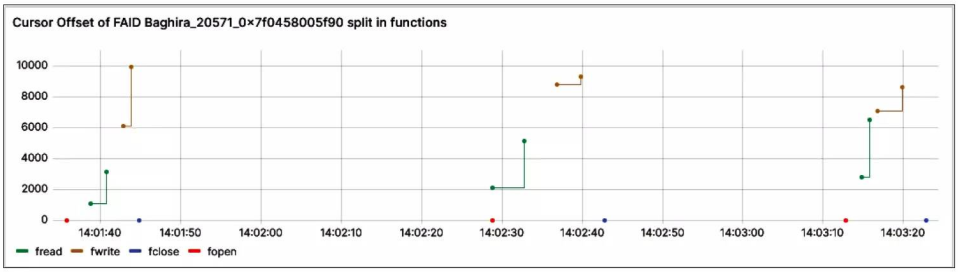


Figure 8: The detailed offset course of the yellow file accesses in Figure 7. The colors represent the called functions.

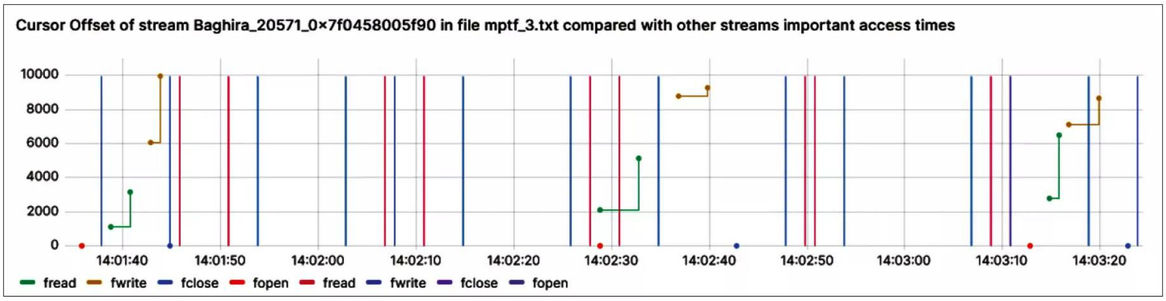


Figure 9: This chart is similar to Figure 7 but has all other relevant function calls of other file accesses displayed as vertical lines.

information about what happened inside the file between the first and last bundled repeated function calls, allowing the script to work with a reduced dataframe.

The script then checks whether each repeated function pair within the bundle can be optimized. To do so, it uses the data in the reduced dataframe to retrieve all function calls on the file from other file accesses during the time between the consecutive function calls.

With this information, the script classifies each pair into self-defined optimization categories.

- Category 0 means it cannot be optimized.
- Category 1 means it can easily be optimized.
- Category 2 means that although one or more other intermediate calls exist, optimization is still possible.
- Category 3 means bail out, because the file access function is not handled.

For each consecutive repeated function call pair, the script creates a new optimization point, regardless of the classification, and adds it to a dataframe containing all analyzed possible optimizations.

An optimization point contains all shared information by both calls, as well as the timestamps of each call and the classified category of the optimization with additional information on the classification reasons. Because this data still is time series data and will be written into the InfluxDB instance to visualize it in Grafana, each point has a timestamp set to the mean of the consecutive functions' timestamps. In this way, you can directly see in a time series diagram to which function calls an optimization point belongs.

To classify each possible optimization and add additional information, the script runs each pair through several checks.

Optimization Classification

The classification process procedes as described earlier in the "Optimization Possibilities" section. Each function pair receives a classification category, as well as an additional information string about the reasons for the categorization.

The first check is whether a function call of a different file access occurs between the checked function pair. If the time-based, filtered, reduced dataframe as described before is empty, no call falls between the pair. In this case, the optimization receives category 1, meaning it can be easily optimized, with the additional information that no calls fall between.

If calls are found between the function pair, further checks have to be made. To check each case efficiently, the script extracts relevant information from the consecutive repeated calls. First, the script retrieves the name of the repeated function, which is the same for both calls in the pair. The only two possible function types here are write and read.

Next, the script determines the byte range inside the file in which both calls occur with the use of the information added earlier about the offset after each function call. The beginning of the affected byte range is the offset before the first function call, which corresponds to the read or write byte count of the first function subtracted from the offset after the first function. The desired information is extracted from the data of the function call. In turn, the end of the affected byte range is one byte before the offset after the second function call.

Lists defined in the script contain all possible function names, including POSIX and MPI function names, that exhibit the same behavior and are treated the same way to ensure that the script handles them equally.

To check how a function in between interferes with the function pair, the script needs the repeated function, as well as the start and end of the affected byte range for the function pair. The previously retrieved list of calls between this pair is iterated, then the script checks each call for its effect on the optimization potential and classifies it accordingly. If one of the calls would prevent an optimization, the loop is broken and the optimization point receives a category 0 or 3 classification, with information given on why it is not possible or cannot be handled.

For each check, the script extracts the relevant details of the current call, including the called function and the range of the affected byte area. Called functions are of type write, read, or seek.

As before, the start of the byte range corresponds to the written or read bytes by the function subtracted from the offset after the function. In the case of a seek function, the start of the byte range is simply the offset after the function call.

The end of the byte range of the call between is one byte before the offset after the function call. If it is a seek function, the byte range is only one point, because a single seek function does not affect a range of bytes. Hence, the start and end of the byte range is the same in such cases.

The script can now compare this information with the details of the repeated functions and performs the checks efficiently.

If the repeated function is a read type, various cases can arise depending on the call in between. In the following cases an optimization is possible: when the call between is another read or a seek, no matter the byte range; when the call between is a write with its byte range ending before or starting behind the read byte range.

Therefore, optimizing repeated read functions is only impossible when the call between is a write interfering with the affected byte area of the read functions. On the other hand, if the repeated functions are of a write type, optimizations are possible when the call between is a read or seek happening in front of or after the writes.

In all other cases, optimizations are impossible because the write functions alter the content in their byte range. Any function between that overlaps with the written byte area prevents possible merging of the repeated writes, because it would lead to inconsistencies when merged.

The mergeable cases are classified category 2, along with information explaining why the call in between would not interfere with merging the repeated functions.

If a function happens to exist that the script cannot handle, it assigns an additional category 3 to it with the information of the unknown function for debugging reasons. This situation is made visible later in the visualization, allowing you to see that it is unclear whether an optimization is possible in these cases.

In any other case, the optimization point receives the default category 0, with information on the problem that prevents a merge.

Either way, the script adds the result as an entry into the earlier described dataframe containing the analyzed optimizations.

After the script analyzes all potential optimizations and the results are in the new dataframe, it writes the data into the InfluxDB instance and sets the category number as the field value of the created points, with all other information added as tags. The timestamp of each point is the mean timestamp of the analyzed function pair.

In the end, the analyzed optimization data is available in the InfluxDB instance, in addition to the offset-enriched file access data, which enables the dashboard to display clearly and effectively when and how optimizations are possible.

Running the Script

When tracing a program with *libiotrace*, the created data has a corresponding job name. The script only requires this job name to identify the data it needs to enrich and analyze. Additionally, it requires a name for the new measurement where the data will be stored in the InfluxDB instance. Adjusting this information before running the script ensures that the desired data is analyzed for optimizations. The script performs the described procedure for each file access within the selected job to analyze potential optimizations then processes each file sequentially, writing all data about cursor position changes and the analyzed optimizations into the InfluxDB instance.

Apart from debugging and additional helpful information, the script also outputs a URL at the end. This URL redirects the user to the Grafana dashboard with all settings preselected according to the analyzed data. The URL automatically selects the variables about the InfluxDB bucket, the measurement containing the data, and the analyzed job name in the dashboard. Also, it sets the time frame for easy use of the dashboard in combination with the script.

Visualization with Grafana

The dashboard comprises two sections in different panels. The dashboard's main goal is to display the program's possible optimizations actively in an intuitive and easy-to-understand way by reducing the information shown to ensure you gain insights without feeling overwhelmed. By setting all the variables and the time frame, the script-generated URL immediately gives you the desired information.

Grafana's dashboard settings at the top are changeable in drop-down boxes, but you do not have to set them manually. This decision was important during the dashboard's design, to support the aim of simplifying its usage. The dashboard is designed to be interactive so that you can adjust the variables. Some panels are clickable, which automatically sets specific variables to show desired information, as described in the following sections.

Flux queries dynamically retrieve the available values for each variable:

```
import "influxdata/influxdb/schema"
schema.tagValues(
  bucket: "${bucket}",
  tag: "_measurement",
  predicate: (r) => true,
  start: -1000d
)
```

This way, the variables automatically include all possible options of the current context without requiring manual updates. Also, some variables and their available options depend on other, sometimes higher level, variables.

The panels display data according to the variable values selected, and as mentioned earlier, you can modify the variables interactively in the panels.

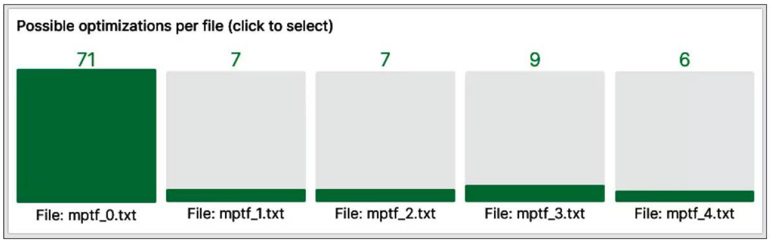


Figure 10: Bar chart displaying number of optimizations found in each file.

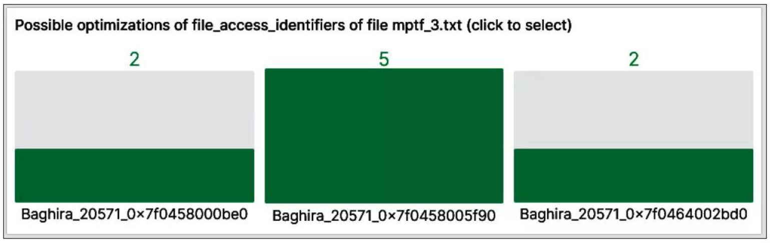
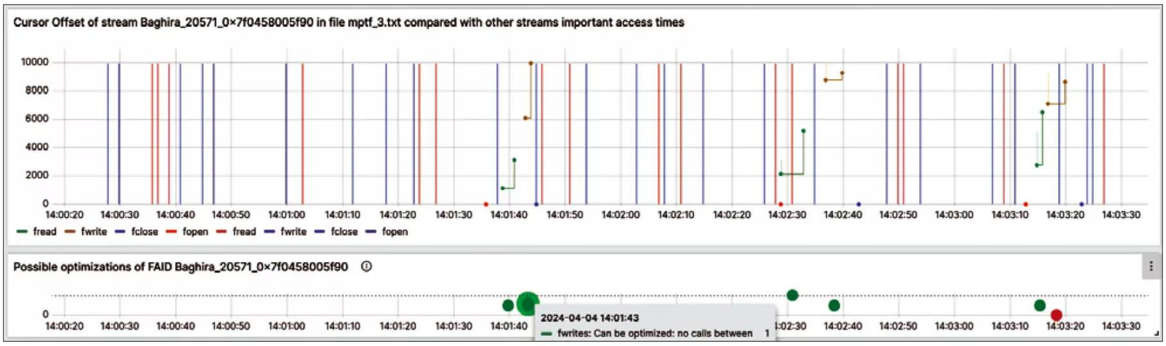


Figure 11: Bar chart displaying the number of optimizations found for each file access of a selected file.



2024-04-04 14:01:43
fwrites: Can be optimized: no calls between 1

Figure 12: Time series panel (top) with calculated optimization points (bottom).

Optimizations Overview

The first section of the dashboard provides an overview of the number and locations of potential optimizations. Vertical bar chart panels visualize their number. The first shows all files used by the program and compares the possible optimizations in each file (Figure 10).

Clicking on any bar automatically selects the clicked file as the value of the corresponding dashboard variable. Data links within the panel enable this type of interaction. These links seamlessly change the desired variables by redirecting to a URL in the same tab, which sets variables according to the clicked field label value:

```
http://localhost:3000/d/<id>/➥
 file-access-analysis?orgId=1&➥
 ${__url_time_range}&${bucket:queryparam}➥
 &${measurement:queryparam}&var-filename=➥
 ${__field.labels.traced_filename}
```

With the selected file automatically set as the variable value, the adjacent bar chart panel displays the number of possible optimizations for all file accesses on this file (Figure 11). This panel is also interactive, so you can select the desired file access by its ID, which automatically updates the corresponding dashboard variable.

The bar charts serve to inform you about the parts of the program that have the highest potential for optimization. Furthermore, these panels provide an easy and interactive way to select the content to view.

Optimization Details

With the file access for investigation selected, the next panels visualize additional information about the file access and its potential optimizations.

The time series panel at the top in Figure 12 shows the detailed course of the file access cursor, along with the calculated offsets. The colors of the points are mapped to the called functions and highlight the consecutive repeated function calls as potential optimization points by automatically connecting the same-colored points.

In addition to the cursor offset of the selected file access, the time series panel displays the function calls of all other file accesses on the same file. To determine whether calls occur between repeating functions, the diagram only shows the times and kinds of functions.

Vertical colored lines indicate the timestamps of the function calls of other file accesses. If a different function call falls between consecutive repeated functions connected by a line, the vertical line visibly crosses the horizontal line, which indicates the potential for optimization where an intermediate function must be considered.

This panel can help you identify where the potential optimizations lie inside the program's code. The analysis of potential optimizations happens inside the script and is displayed in the panel beneath the time series panel.

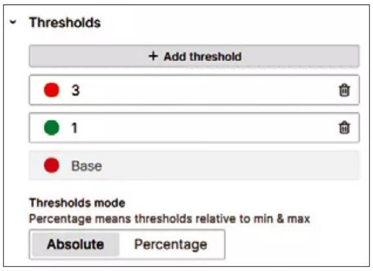
The optimization diagram in Figure 12 visualizes all optimization points, whether possible or not, and is aligned to the panel above, so you can see directly which optimization point corresponds to which repeated function calls. The color coding of the points indicate their classification.

Points with the value and classification 0 cannot be optimized and appear in red.

The panel shows points with values 1 and 2 in green, indicating that the cases can be optimized. If a point has value 3 and contains a function that cannot be handled, it is orange.

This color coding is realized with the *Thresholds* panel option (Figure 13).

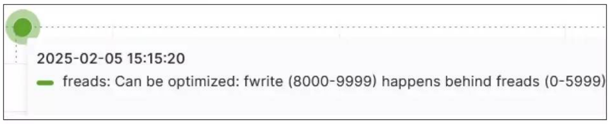
Mousing over these points reveals information that explains their classification (Figure 14).



Thresholds
🟥 3
🟩 1
🟥 Base

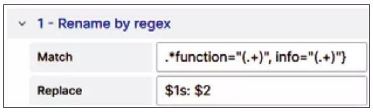
Thresholds mode
Percentage means thresholds relative to min & max
Absolute | Percentage

Figure 13: The panel option required for color coding optimization categories.



2025-02-05 15:15:20
freads: Can be optimized: fwrite (8000-9999) happens behind freads (0-5999)

Figure 14: Information about the reason for the classification is displayed when mousing over an optimization point.



1 - Rename by regex

Match .*function="(.+)", info="(.+)"}

Replace $1s: $2

Figure 15: Option to rename points that display relevant information.

The retrieved data points have a specific structure, containing information added by the script as tags for each point:

```
{_start="<start-timestamp>", _stop=↩
 "<stop-timestamp>", function="fread",↩
 info="Can be optimized: fwrite..."}
```

To display only the relevant information on mouse-over, the panel transforms the data with the *Rename by regex* transformation, extracting only the values of the required tags (Figure 15). In this way, you receive the desired information as needed and can see directly how many and where optimizations are possible.

Insights Gained by the Tool

Too many small I/O function calls add unnecessary overhead. In summary, the pipeline works as follows: Monitor and trace a program during runtime with *libiotrace*, run the Python script working with the created data, and receive the desired insights in the Grafana dashboard. The dashboard explains the optimization possibilities, giving additional information that helps you understand each case. Applying those optimizations causes more efficient file access patterns in the program and thus higher performance, running quicker and using fewer resources.

To prove that this work can improve a program's efficiency, we created a small program in C simple enough to repeat that simulated several file accesses with `fwrite`, `fread`, and `fseek` function calls. By running the compiled program with active *libiotrace* measurements and by using the script presented here, we could view the optimization analysis in the Grafana dashboard. The execution time of the program was also measured.

After optimizing the script with the help of the dashboard suggestions and executing the program again, we saw that it performed drastically fewer function calls because calls were merged when possible.

When comparing the mean execution times of 10 program executions of the optimized and unoptimized versions, a 4.15 percent improvement in overall time was observed. The tested program's execution time was a little more than three seconds.

Applying this method to much bigger programs can lead to much greater improvements, affecting not only time consumption, but also resource and energy usage in the long term. ■

Info

[1] Darshan: [https://www.mcs.anl.gov/research/projects/darshan/]

[2] Vampir: [https://vampir.eu/]

[3] libiotrace: [https://github.com/hpcraink/fsprj2], [https://www.admin-magazine.com/Archive/2022/71/A-ptrace-based-tracing-mechanism-for-syscalls]

[4] POSIX: [https://pubs.opengroup.org/onlinepubs/9699919799]

[5] MPI: [https://www.mpi-forum.org]

[6] Script described in this article: [https://linuxnewmedia.thegood.cloud/s/9nFQcFb2p8oRMEJ]

[7] pandas.DataFrame: [https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html]

The Authors

Andreas Scheumann is currently pursuing a master's degree in Applied Computer Sciences at the Hochschule Esslingen (University of Applied Sciences, Esslingen am Neckar, Germany) while working as a cloud specialist. **Philipp Koester** is pursuing his PhD at the Hochschule Esslingen and the High-Performance Computing Center (HLRS, Stuttgart). **Rainer Keller** is a full Professor at Hochschule Esslingen of Computer Sciences specializing in high-performance computing.

■

JAX LONDON

The Conference for
Java and Software Innovation

October 6 – 9, 2025
London or online

BOOST YOUR JAVA SKILLS

JAX has been providing valuable, practical know-how about the world's most popular programming language. Learn from leading experts how to make your Java projects a success.

EXPERIENCE SOFTWARE ARCHITECTURE

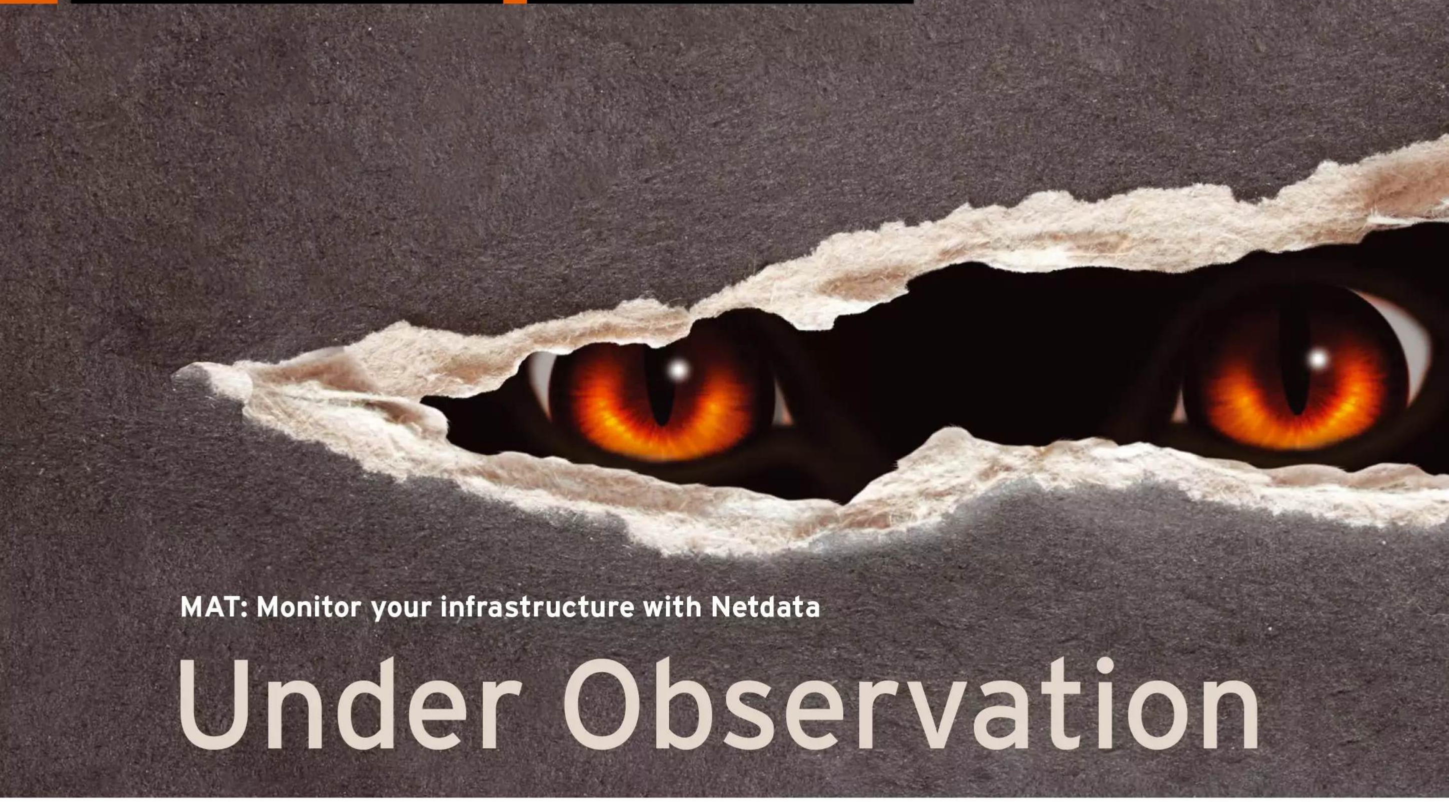
Enterprise Java, Spring, Microservices, DevOps, Cloud – find out how technological innovations can be integrated into overall contexts to build sustainable and future-proof IT systems.

UNDERSTAND TRENDS

What is behind the current trends like JavaScript, machine learning, blockchain & serverless? Understand the background and learn how to implement innovations into your projects.

Presented by: devmio | Organizer: S&S Media Group

jaxlondon.com



MAT: Monitor your infrastructure with Netdata

Under Observation

Netdata relies on easy commissioning and a hosted cloud service to simplify monitoring, alerting, and trending. More than 300 services can be monitored in addition to basic system parameters. By Martin Gerhard Loschwitz

Monitoring, alerting, and trending (MAT) are rarely at the top of the list of important topics for admins, but they simply cannot be avoided in everyday life. Experienced admins in particular will be familiar with the problem of Nagios and its numerous forks and offshoots, which used to be obvious choices for the prevailing IT landscape, being difficult to transfer later to containerized microservice architectures.

In the meantime, much has happened in the monitoring world. Increasing numbers of solutions are capable of automatically identifying targets and adding them to their configurations. Instead of legacy event monitoring, trending is seeing more frequent use, to the extent that event monitoring is becoming a kind of byproduct. However, those who have ever worked with Prometheus and had to develop a strategy for the long-term storage of metrics data will not relish the prospect.

Netdata promises to do things better. In this article, I take a closer look at how Netdata works to discover exactly what the solution offers.

Netdata

With a remarkably simple architecture, Netdata (Figure 1) lets you monitor your infrastructure without too much overhead. To do this, you just need to install the Netdata agent on the devices in your infrastructure and set up a connection to Netdata Cloud to create a complete monitoring system quickly with a colorful graphical user interface (GUI), functional display, and meaningful alerting.

If you think Netdata is a commercial provider, you are partly right. The complete software is available under a free license. Although a charge is levied for using the Netdata cloud service, the price spectrum will not cause experienced admins to break a sweat. The bottom line is also significantly cheaper than employing one or more full-timers to operate and maintain your in-house monitoring solution.

Licensing Questions

The Netdata website [1] is full of superlatives, with talk of infinite scalability, straightforward pricing, and maximum cost efficiency. The whole thing gives the impression that you are dealing with a Silicon Valley start-up marketing a proprietary monitoring solution. At least in terms of the location, you're not entirely wrong: Netdata Inc. is based in San Francisco, and as you might expect, Netdata is also backed by major investors.

However, the bit about proprietary software is not quite right. Although Netdata is a little circumscript when explaining the license model, it is basically quite simple. The Netdata agent, which runs directly on the customer systems and accesses the data of the services to be monitored there, is licensed under version 3 of the GNU General Public License (GPLv3) and is therefore completely open software. The provider also explicitly points this out, maybe to dispel security concerns up front.

The situation is different with two other components: Netdata Cloud and Netdata UI. These components make the whole setup really interesting. Although the Netdata agent can theoretically run locally without a connection to Netdata Cloud and Netdata UI, in practical terms the two components handle all analysis tasks and prepare the acquired data for dashboards and trending graphs (Figure 2). Unfortunately, neither of the two components is under an open license; they are proprietary software.

If you do not want to use Netdata in the cloud, you still have the option

Lead Image © Natalia Lukiyanova, 123RF.com

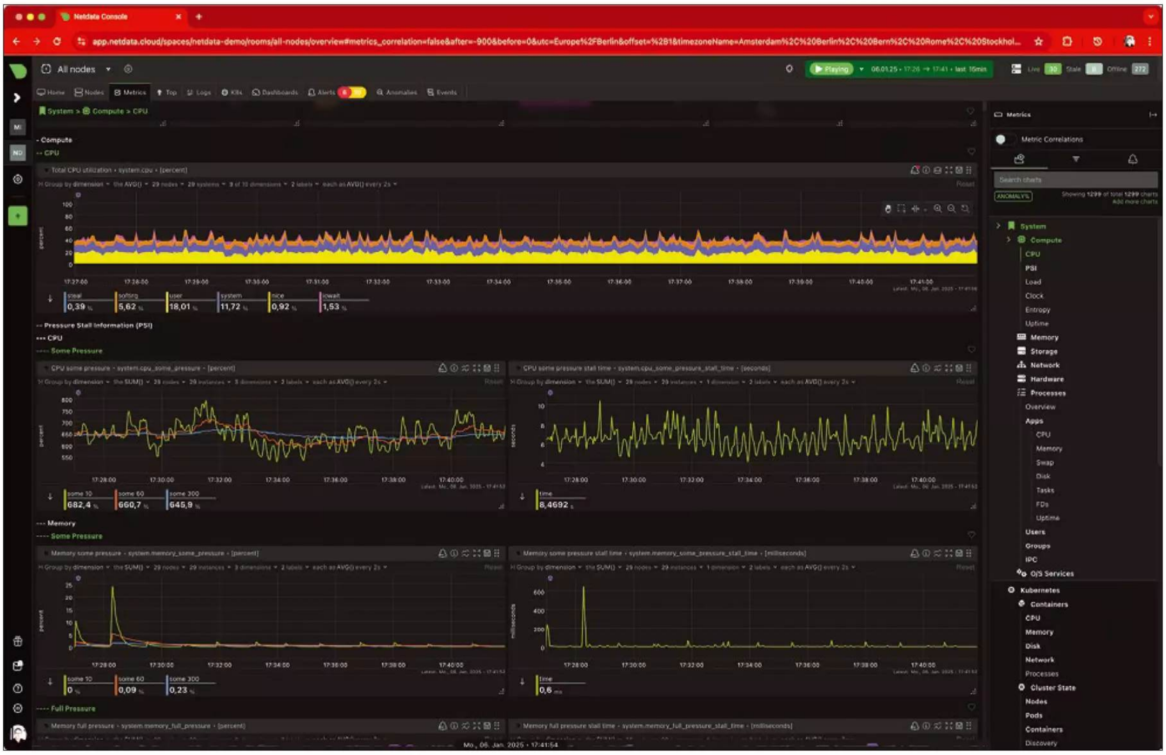
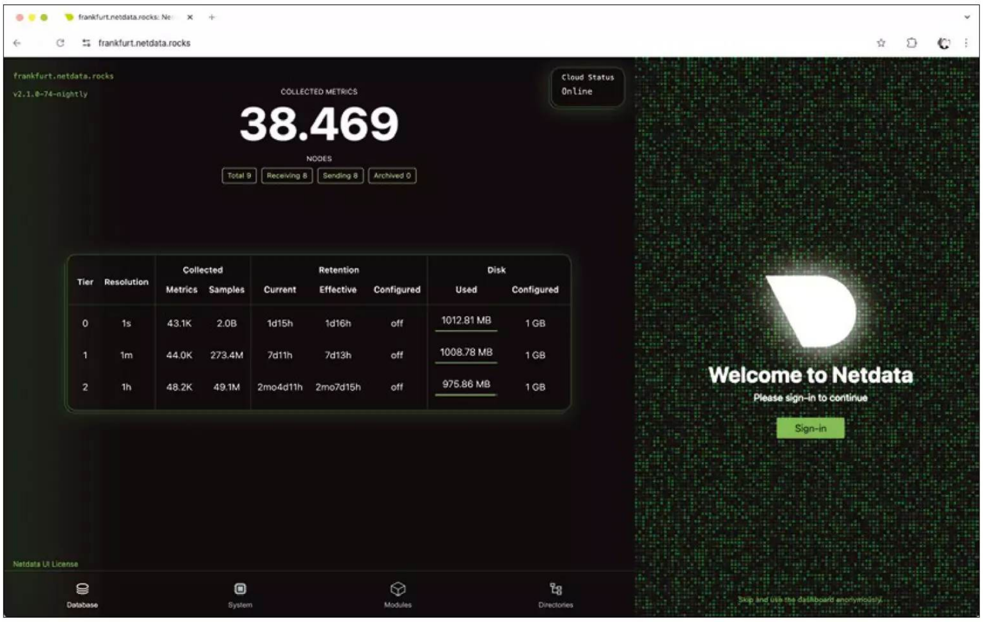

Netdata Console
All nodes
Home
Nodes
Metrics
Top
Logs
K8s
Dashboards
Alerts
Anomalies
Events
System > Compute > CPU
Compute
CPU
Total CPU utilization · system.cpu · [percent]
steal 0,39
softirq 5,62
user 18,01
system 11,72
nice 0,92
iowait 1,53
Pressure Stall Information (PSI)
CPU
Some Pressure
some 10 682,4
some 60 660,7
some 300 645,9
time 8,4692
Memory
Some Pressure
some 10 0
some 60 0,09
some 300 0,23
time 0,6
Full Pressure
Metrics
Metric Correlations
System
Compute
CPU
PSI
Load
Clock
Entropy
Uptime
Memory
Storage
Network
Hardware
Processes
Overview
Apps
Users
Groups
IPC
O/S Services
Kubernetes
Containers
Cluster State
Nodes
Pods
Discovery
Mo., 06. Jan. 2025 · 17:41:54

Figure 1: Netdata offers comprehensive monitoring for services on premises and in the cloud. It generates clear user interface (UI) views from the collected metrics data.

of installing it on premises, which certainly dispels European concerns about the General Data Protection Regulation (GDPR), but to refer to Netdata as a completely open source and free solution would be an exaggeration. The license model structure implicitly explains how Netdata, or rather its architecture, works. Customers roll out the agent on their infrastructure and specify the desired Netdata cloud platform as the target. Data acquisition starts, and almost immediately, Netdata Cloud and Netdata UI return colorful charts and assume the tasks of monitoring the services on your target systems, some of which will be detected automatically.


frankfurt.netdata.rocks
frankfurt.netdata.rocks
v2.1.0-74-nightly
COLLECTED METRICS
38.469
NODES
Total 9
Receiving 8
Sending 8
Archived 0
Cloud Status
Online
Welcome to Netdata
Please sign-in to continue
Sign-in
Netdata UI License
Database
System
Modules
Directories
Skip and use the dashboard anonymously

Tier	Resolution	Collected		Retention			Disk	
		Metrics	Samples	Current	Effective	Configured	Used	Configured
0	1s	43.1K	2.0B	1d15h	1d16h	off	1012.81 MB	1 GB
1	1m	44.0K	273.4M	7d11h	7d13h	off	1008.78 MB	1 GB
2	1h	48.2K	49.1M	2mo4d11h	2mo7d15h	off	975.86 MB	1 GB

Figure 2: Netdata has several cloud endpoints worldwide. Alternatively, you can operate the product on premises.

What Netdata Monitors

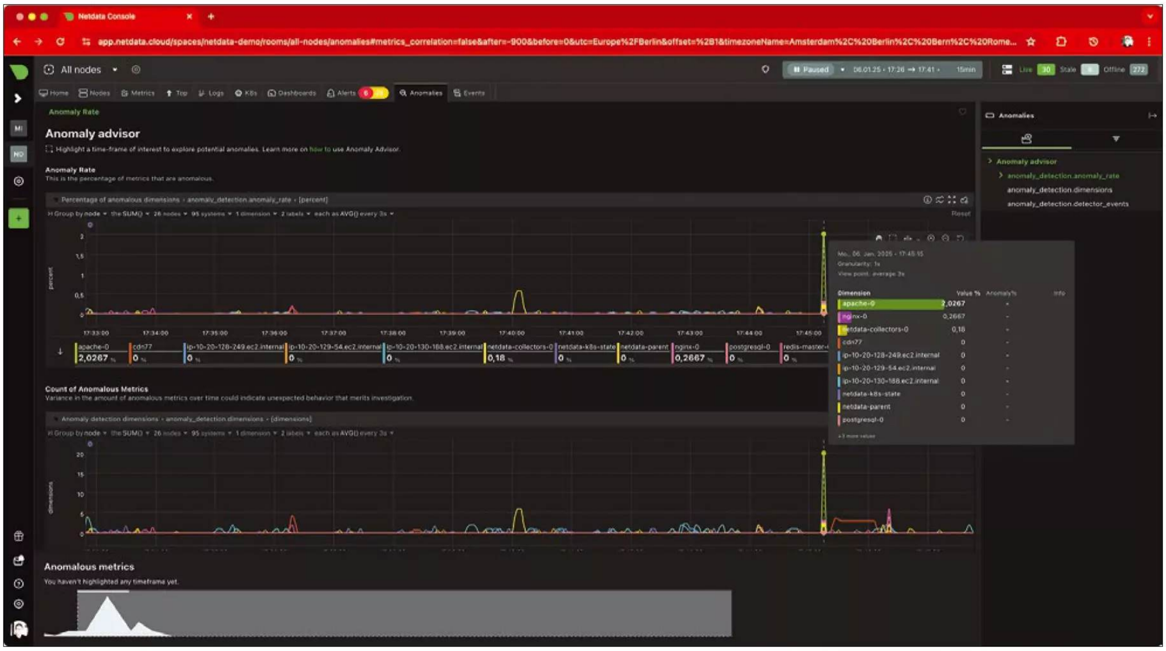
From the point of view of an organization that needs a simple and functional monitoring system, the license issue could be of secondary interest. In this case, practical aspects are probably more important, such as the question of which targets Netdata can monitor and how it does so. The Netdata developers claim more than 300 supported monitoring targets, which does not even include the basic metrics of individual systems such as CPU and RAM utilization, I/O, network performance, and so on.

Basically, Netdata acts more like a time series database than a classic monitoring system, which makes its functionality more similar to that of Prometheus than Zabbix or Icinga.

The agent collects data on basic system metrics and specific services such as MySQL on the target systems. From this data, Netdata then draws conclusions about the status of the services and raises an alarm, if necessary. It would be well beyond the scope of this article to look at every supported target service at this point. The highlights include the classic Linux standard services for authorization and authentication (i.e., OpenLDAP, SSH, FreeRADIUS, HashiCorp Vault, etc.). Netdata also covers classic network infrastructure services, including BIND, PowerDNS, most DHCP servers, and common databases (e.g., PostgreSQL, MySQL, and MariaDB). Netdata can seamlessly monitor NTP and Chrony as time servers without any additional configuration. Systemd gets a completely separate section: Netdata monitors systemd's metrics in a practically seamless way, including various details on top, such as the total number of services started or the number of services that systemd has tagged as *Failed*.

If you are looking for a comprehensive way of monitoring your hardware, Netdata can do so. The software can read out most common hardware sensors to aggregate data. This capability proves to be very helpful in various places (e.g., temperature sensors in servers). If a case fan dies, Netdata notices – on one hand because it records the data from the fan itself, but on the other hand (presumably) because the temperature in the server increases. Hardware sensors can also be controlled and read outside of the classic IT infrastructure. For example, if you were to install thermometers in your home as part of your smart home setup, you could use Netdata to monitor them. One particularly practical aspect is that Netdata does not necessarily have to rely on the Linux kernel or its drivers in servers to discover the values of the various sensors. It can read these values directly by HP Integrated Lights-Out (iLO), IBM Rational Software Architect (RSA), and Dell Remote Access Controller (DRAC) or generically with the Intelligent Platform Management Interface (IPMI) or Redfish. When it comes to the Internet of things (IoT), Netdata also leverages the full potential: The service can connect to practically all popular open source solutions for home automation, working directly with OpenHAB, with connectors for adapters such as Homebridge, which can be connected to other solutions such as the FHEM house automation system.

If you frequently deal with smart home scenarios, you will already be familiar with the zoo of different standards: You will appreciate Netdata's versatility. In addition to collection services, Netdata can communicate directly with various IoT tools from established manufacturers. API tokens can connect to Philips Hue and devices from Netatmo and Elgato. If you operate a photovoltaic or battery


Netdata Console
app.netdata.cloud/spaces/netdata-demo/rooms/all-nodes/anomalies#metrics_correlation=false&after=-900&before=0&utc=Europe%2FBerlin&offset=%2B1&timezoneName=Amsterdam%2C%20Berlin%2C%20Bern%2C%20Rome...
All nodes
Paused
06.01.25 • 17:26 → 17:41 •
15min
Live 30
Stale
Offline 272
Home
Nodes
Metrics
Top
Logs
K8s
Dashboards
Alerts
Anomalies
Events
Anomaly Rate
Anomaly advisor
Anomalies
Anomaly advisor
anomaly_detection.anomaly_rate
anomaly_detection.dimensions
anomaly_detection.detector_events
Percentage of anomalous dimensions • anomaly_detection.anomaly_rate • [percent]
Group by node • the SUM() • 26 nodes • 95 systems • 1 dimension • 2 labels • each as AVG() every 3s
Reset
percent
Dimension
Value %
Anomaly%
Info
apache-0
2,0267
nginx-0
0,2667
netdata-collectors-0
0,18
Count of Anomalous Metrics
Anomaly detection dimensions • anomaly_detection.dimensions • [dimensions]
dimensions
Anomalous metrics

Highlight a time-frame of interest to explore potential anomalies. Learn more on how to use Anomaly Advisor.

Anomaly Rate
This is the percentage of metrics that are anomalous.

Count of Anomalous Metrics
Variance in the amount of anomalous metrics over time could indicate unexpected behavior that merits investigation.

Anomalous metrics
You haven't highlighted any timeframe yet.

Figure 3: Netdata uses eBPF to provide AI-supported anomaly detection for network traffic. In doing so, it is following in the footsteps of services such as Coroot.

storage system (e.g., SolarEdge hardware or SMA converters from SMA Solar Technology AG), you also have a good chance of accessing the metrics data they generate. In addition to acquisition tools for Linux, FreeBSD and macOS counterparts exist.

The Netdata developers clearly and carefully consider the source of their data. They have written a dedicated module for importing metric data from eBPF, Linux's in-kernel virtual machine that allows arbitrary intervention in network traffic. Armed with suitable virtual eBPF instances, practically the entire status of a network can be read out and interpreted. Coroot [2] is another monitoring solution that leverages this ability, but the Netdata programmers take things even further than Coroot with an artificial intelligence (AI) function on the Netdata dashboard designed to detect anomalies in network traffic on the basis of data from eBGP (Figure 3).

Kubernetes Support

In 2025, a monitoring solution without support for Kubernetes would be a strange thing. Reassuringly, the Netdata developers now offer a containerized version of their agent (Figure 4) and ensure the appropriate Kubernetes integration.

If you primarily operate your setups in the context of containers and in conjunction with Kubernetes, you are likely to benefit from various features. For example, the Netdata agent in Kubernetes can be controlled directly as a native resource over the Kubernetes API, which makes it possible to roll the agent out as part of a network along with other applications. Moreover, Netdata can correlate the collected metrics data with the Kubernetes metadata in this way.

It would not be very helpful just to know that an instance of MySQL started by Kubernetes (K8s) is running on some system, because it would not give you any insights into which specific workload the instance belongs. To counter this, Netdata also docks to the K8s API and reads the metadata for each monitored container. In this way, Netdata can display a coherent view in the user interface, providing information on both the service itself and the Kubernetes context in which it is running (Figure 5).

However, the developers go even further by adding support in Netdata for extensions that are often used in the Kubernetes context. For example, Cilium [3] is considered one of the most comprehensive software-defined network (SDN) solutions for K8s. Netdata can seamlessly monitor the Cilium agent and the Cilium operator in K8s as well as the Cilium proxy. Also, Netdata lets you check the Rancher Kubernetes manager.

Although K8s is already considered a cloud in many places, it's a different story when it comes to classic infrastructure-as-a-service (IaaS) platforms or hyperscalers. Netdata offers comprehensive integration for AWS, Azure, and Google Cloud Platform (GCP) and support for several private cloud products. If you use Akamai's services as a content delivery network (CDN), for example, you dock Netdata onto the API to field most of the metrics data. The same applies to practically all standard services in AWS, including Amazon

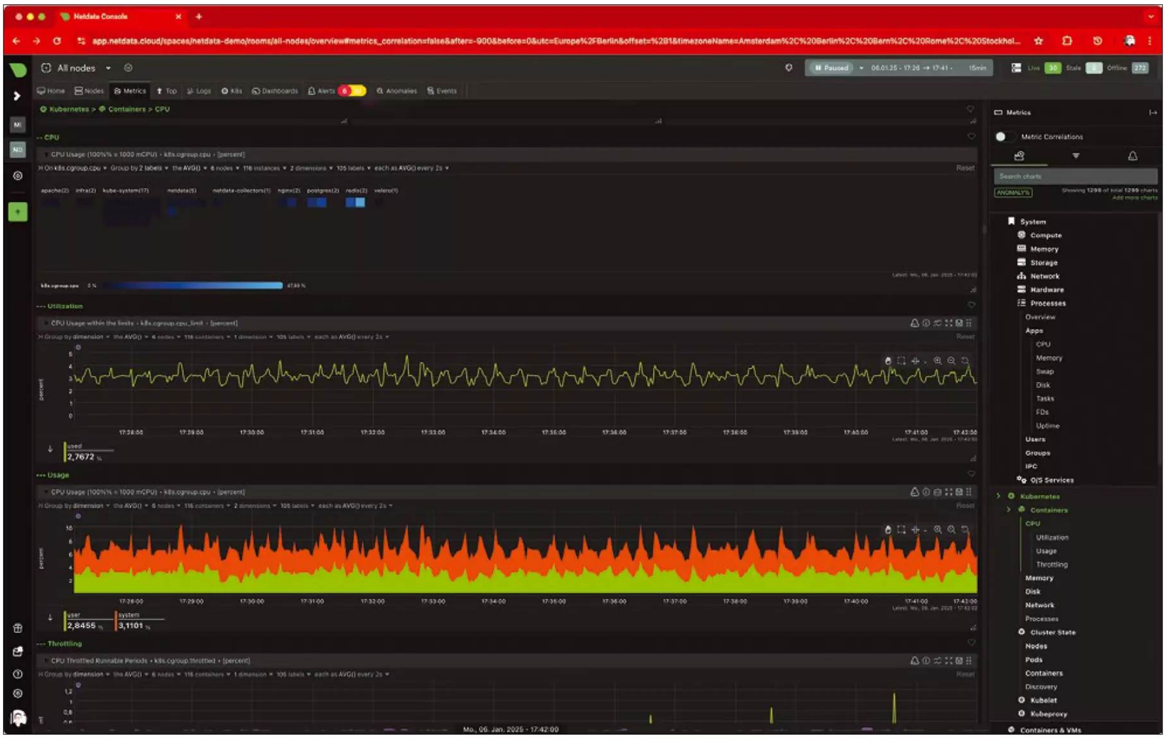

Netdata Console
app.netdata.cloud/spaces/netdata-demo/rooms/all-nodes/overview#metrics_correlation=false&after=-900&before=0&utc=Europe%2FBerlin&offset=%2B1&timezoneName=Amsterdam%2C%20Berlin%2C%20Bern%2C%20Rome%2C%20Stockhol...
All nodes
Home
Nodes
Metrics
Top
Logs
K8s
Dashboards
Alerts
Anomalies
Events
Paused
15min
Live 30
Stale
Offline 272
Kubernetes > Containers > CPU
CPU
CPU Usage (100%% = 1000 mCPU) · k8s.cgroup.cpu · [percent]
apache(2) infra(2) kube-system(17) netdata(5) netdata-collectors(1) nginx(2) postgres(2) redis(2) velero(1)
Utilization
CPU Usage within the limits · k8s.cgroup.cpu_limit · [percent]
used 2,7672 %
Usage
CPU Usage (100%% = 1000 mCPU) · k8s.cgroup.cpu · [percent]
user 2,8455 %
system 3,1101 %
Throttling
CPU Throttled Runnable Periods · k8s.cgroup.throttled · [percent]
Mo., 06. Jan. 2025 · 17:42:00
Reset
Metrics
Metric Correlations
Search charts
ANOMALY%
Showing 1299 of total 1299 charts
Add more charts
System
Compute
Memory
Storage
Network
Hardware
Processes
Overview
Apps
CPU
Memory
Swap
Disk
Tasks
FDs
Uptime
Users
Groups
IPC
O/S Services
Kubernetes
Containers
CPU
Utilization
Usage
Throttling
Memory
Disk
Network
Processes
Cluster State
Nodes
Pods
Containers
Discovery
Kubelet
Kubeproxy
Containers & VMs

Figure 4: Netdata records the metrics of individual processes, as well as the data of active containers.

Elastic Compute Cloud (EC2), Elastic Container Service (ECS), and Simple Storage Service (S3). Netdata can also handle basic Azure resources. If you operate Nextcloud, you can monitor it with Netdata just as seamlessly as you can the OpenStack private cloud environment.

Chatty Tool

If you are looking to deploy Netdata in your setup, you will need to decide whether or not you really want to put all of your metrics data in Netdata's hands. The developers bandy superlatives about here, too: *Infinite aggregation*, for example, is claimed by nesting multiple Netdata agents. Individual instances on the target systems first send their data to intermediary agents, which then communicate with Netdata Cloud. Because any number of incarnations of each element in this construct can exist at any level, the whole enchilada will scale infinitely, at least in theory.

Unfortunately, the proprietary Netdata dashboard and Netdata Cloud components make you dependent on the manufacturer. If you want to avoid this vendor lock-in, you will need to store your metadata locally besides storing in the Netdata Cloud. It is to the credit of the developers that they provide comprehensive interfaces in their agent for this purpose. For example, if you run a local Prometheus instance parallel to Netdata and want to send the metrics data from the installation to the Netdata Cloud at the same time, you can simply configure the appropriate Prometheus endpoint in the Netdata agent. The same applies to other well-known time series databases such as InfluxDB, Gnocchi, M3, and OpenTSDB.

VictoriaMetrics and Thanos are also supported target platforms. The Netdata agent uses the Prometheus interface, which VictoriaMetrics and Thanos both support; the same applies to Cortex. If Prometheus is your preferred time series database, sooner or later you will need M3, Cortex, or Thanos [4] anyway; otherwise, you will just be cluttering your local Prometheus setup with metrics data, and experience shows that this will slow down Prometheus over time. Either way, anyone who uses a reasonably up-to-date time series database will have no difficulty exporting their data to Netdata. If you decide against Netdata at some later stage, you will still have the data available for use in Grafana or another user interface.

Performance Questions

Anyone who has lived with Prometheus in the recent past might also have experienced its temporary performance bottlenecks. Large instances of the service in particular often become unbearably slow, and that doesn't even take into account the load on the monitored systems.

Prometheus also relies on a server-agent architecture to collect metrics data, although many target services have separate agents, known as exporters. Depending on the data collection type and scope, some Prometheus exporters can take quite a big bite out of RAM and compute resources, none of which detracts from Prometheus's popularity and widespread use.

This is probably why the Netdata developers chose it as the target for a direct comparison with their software. The figures provided by Netdata speak for themselves: The Netdata agent, as a single tool trimmed for efficiency, is said to generate an average of 40 percent less CPU load and use a maximum of just over half

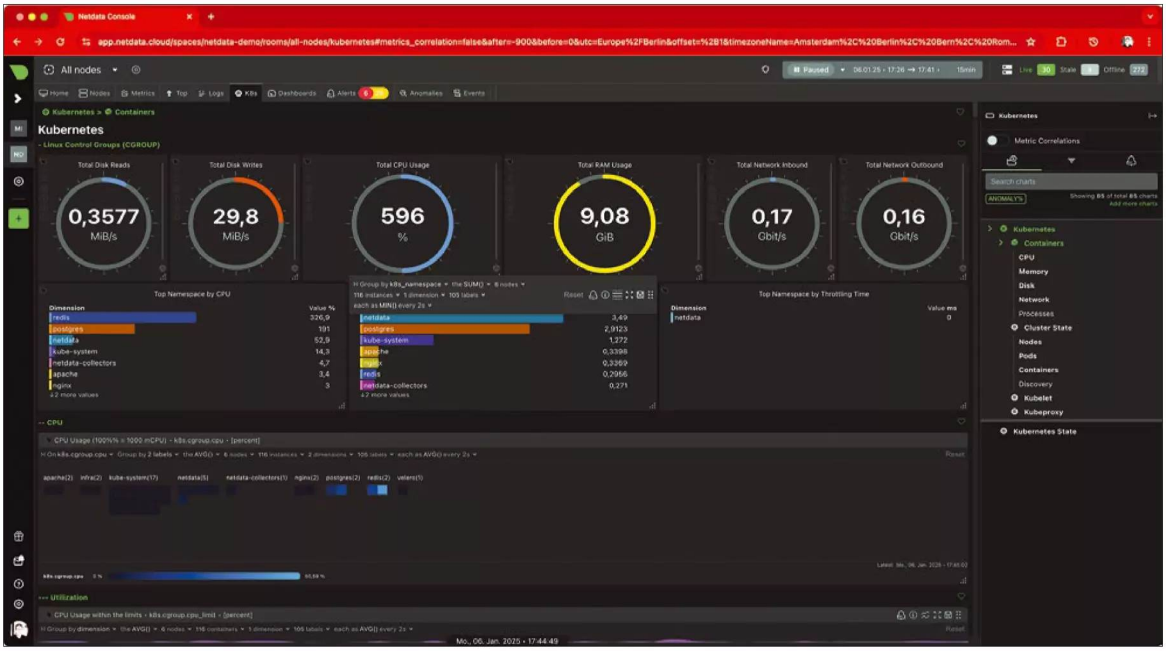
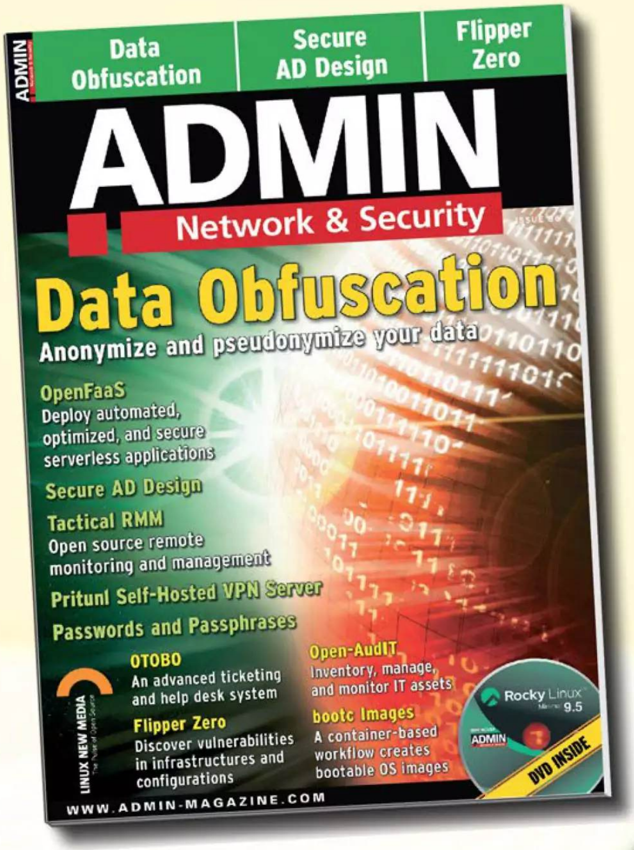

Netdata Console
All nodes
Kubernetes
Total Disk Reads
0,3577
MiB/s
Total Disk Writes
29,8
MiB/s
Total CPU Usage
596
%
Total RAM Usage
9,08
GiB
Total Network Inbound
0,17
Gbit/s
Total Network Outbound
0,16
Gbit/s
Top Namespace by CPU
Top Namespace by Throttling Time
Metric Correlations
Kubernetes
Containers
CPU
Memory
Disk
Network
Cluster State
Nodes
Pods
Containers
Kubelet
Kubeproxy
Kubernetes State

Figure 5: Netdata connects directly to the Kubernetes API and reads the metrics data available there before displaying the information in the GUI.

REAL SOLUTIONS FOR REAL NETWORKS



Data Obfuscation | Secure AD Design | Flipper Zero

ADMIN

Network & Security

Data Obfuscation

Anonymize and pseudonymize your data

OpenFaaS
Deploy automated, optimized, and secure serverless applications

Secure AD Design

Tactical RMM
Open source remote monitoring and management

Pritunl Self-Hosted VPN Server

Passwords and Passphrases

OTOBO
An advanced ticketing and help desk system

Flipper Zero
Discover vulnerabilities in infrastructures and configurations

Open-AudIT
Inventory, manage, and monitor IT assets

bootc Images
A container-based workflow creates bootable OS images

Rocky Linux 9.5

DVD INSIDE

LINUX NEW MEDIA

WWW.ADMIN-MAGAZINE.COM

ADMIN is your source for technical solutions to real-world problems.

Improve your admin skills with practical articles on:

- Security
- Cloud computing
- DevOps
- HPC
- Storage and more!

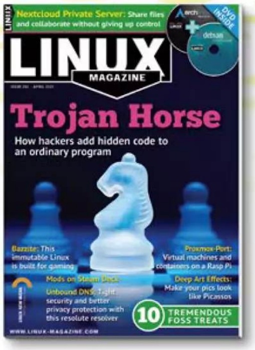
SUBSCRIBE NOW!

shop.linuxnewmedia.com/shop

ADMIN magazine

@adminmag

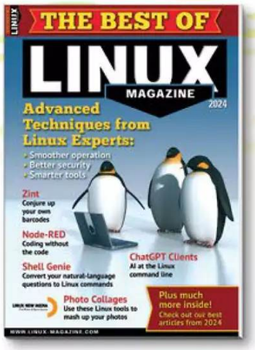
@adminmagazine
@hachyderm.io



LINUX MAGAZINE

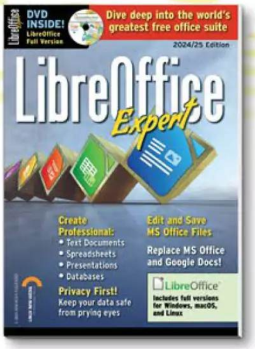
Trojan Horse

How hackers add hidden code to an ordinary program



THE BEST OF LINUX MAGAZINE 2024

Advanced Techniques from Linux Experts:



Dive deep into the world's greatest free office suite

LibreOffice Expert

RASP PI TRICKS

MakerSpace #04

HANDS-ON PROJECTS FOR MAKERS

CHECK OUT OUR FULL CATALOG: SHOP.LINUXNEWMEDIA.COM

the available RAM. Mind you, this is based on the premise of Netdata storing almost three days' more metrics data than Prometheus in the comparison.

Moreover, Netdata supports a significantly higher sampling rate. By default, the agent reads the configured metrics data once per second, whereas most admins tend to set the Prometheus query interval to 20 seconds. Objectively, Netdata achieves a significantly higher level of detail in its metrics data, although not every setup will benefit.

Log Aggregation

In the context of state-of-the-art system monitoring, a long-established mantra states that both the metrics data and the log messages must be part of the dataset to be monitored, which only makes sense: The data gives you detailed information on errors that occur. The crux of the matter is that logs are still not standardized and therefore not machine-readable.

The field of log aggregation has been hotly contested on the market for more than 10 years. Splunk [5] is considered to be the king of the hill. No other solution is capable of reading and indexing log messages as comprehensively as this product. That said, the vendor extracts a high price for all this functionality. Although you can find alternatives such as the ELK stack (Elasticsearch, Logstash, Kibana) or Grafana Loki, they are not yet very widespread.

Netdata chooses a different approach: For more than a year, the manufacturer has quietly added various log aggregation functions to its tool. The messages become part of the metrics data that the agent sends to Netdata Cloud. The service evaluates these messages immediately and makes them part of regular monitoring. Anyone who uses Netdata's log aggregation feature therefore has access to the same functions for logs as for the metrics data itself, which makes it possible to search for specific details and alert on the basis of specific log entries.

Netdata and systemd integration is of central importance, of course. Netdata continuously reads the systemd journal, which is available on practically every recent Linux system, and forwards the details to Netdata Cloud. A file agent can also be used to collect the contents of text files. Netdata also automatically collects the logs from various services that are monitored anyway, unless you disable it. Once the log aggregation function has been activated, and because Netdata is monitoring MariaDB anyway, the agent also transmits the matching logs, which keeps the configuration overhead to a minimum.

Comprehensive Alerting

The best monitoring tools are useless if problems do not trigger alerts, and Netdata leaves nothing to chance. It comes with a pre-configured alerting setup for practically every collector (i.e., for the modules that aggregate the metrics data of a specific service) according to the various manufacturers' best practices and standards.

As an admin, it is then up to you to configure the Netdata Cloud alerting module so that you receive the alerts. The text messages that used to be commonplace are no longer the norm in increasing numbers of organizations. Instead, the focus is on cloud services and Netdata's capabilities are again quite impressive. It can contact the major US services (e.g., Opsgenie and PagerDuty, but also Discord, Mattermost, Rocket.Chat, Splunk, and Telegram). If you do want to send a short message, you can use a webhook and connect it to your SMS provider's API. If you don't want to use a centralized cloud setup for alerting, you can tell the Netdata agent to send the alerts directly. Even more contact options than with the cloud version are available, including good old IRC, email, and short message.

All told, Netdata helps you implement comprehensive alerting quickly, not least because the service's default settings are meaningful. In some situations, Netdata's developers have weighed up whether or not an alert is necessary (e.g., if one of dozens of web servers in a web server setup fails). The default configuration avoids an alert being sent out in the middle of the night, although you can adjust this setting if so desired.

Conclusions

Two things stand out when working with Netdata: The program launches incredibly quickly and works extremely well without further need for configuration. Without reservation, the developers keep the promise that virtually no other solution can produce usable results so quickly.

One drawback is that Netdata has to be operated as a cloud service with proprietary components, although you can run Netdata Cloud on your own infrastructure. The $10,000 or so that an on-premises setup on this scale costs [6] is probably not an issue for most organizations.

More serious is that every organization that uses Netdata has to accept vendor lock-in for part of its infrastructure. If this condition does not worry you, you will want to put Netdata on your shopping list, especially if you are looking to replace or at least upgrade an outmoded monitoring system.

Info

[1] Netdata: [https://www.netdata.cloud]

[2] "Intelligent observability with AI and Coroot" by Martin Gerhard Loschwitz, *ADMIN*, issue 87, 2025 (this issue)

[3] Cilium: [https://cilium.io]

[4] "Four Solutions for Prometheus Long-Term Storage" by Martin Gerhard Loschwitz, *ADMIN*, issue 85, 2025, pg. 24, [https://www.admin-magazine.com/Archive/2025/85/Four-solutions-for-Prometheus-long-term-storage/]

[5] Splunk: [https://www.splunk.com]

[6] Netdata pricing: [https://www.netdata.cloud/pricing/]

The Author

Martin Loschwitz is the founder and managing director of True West IT Services GmbH, which offers scalable IT infrastructure based on OpenStack and Kubernetes.

Looking for
Open Source talent?

Dedicated to professionals in the open source industry, Open Source JobHub will help you connect with qualified candidates.

OpenSource
JOB HUB

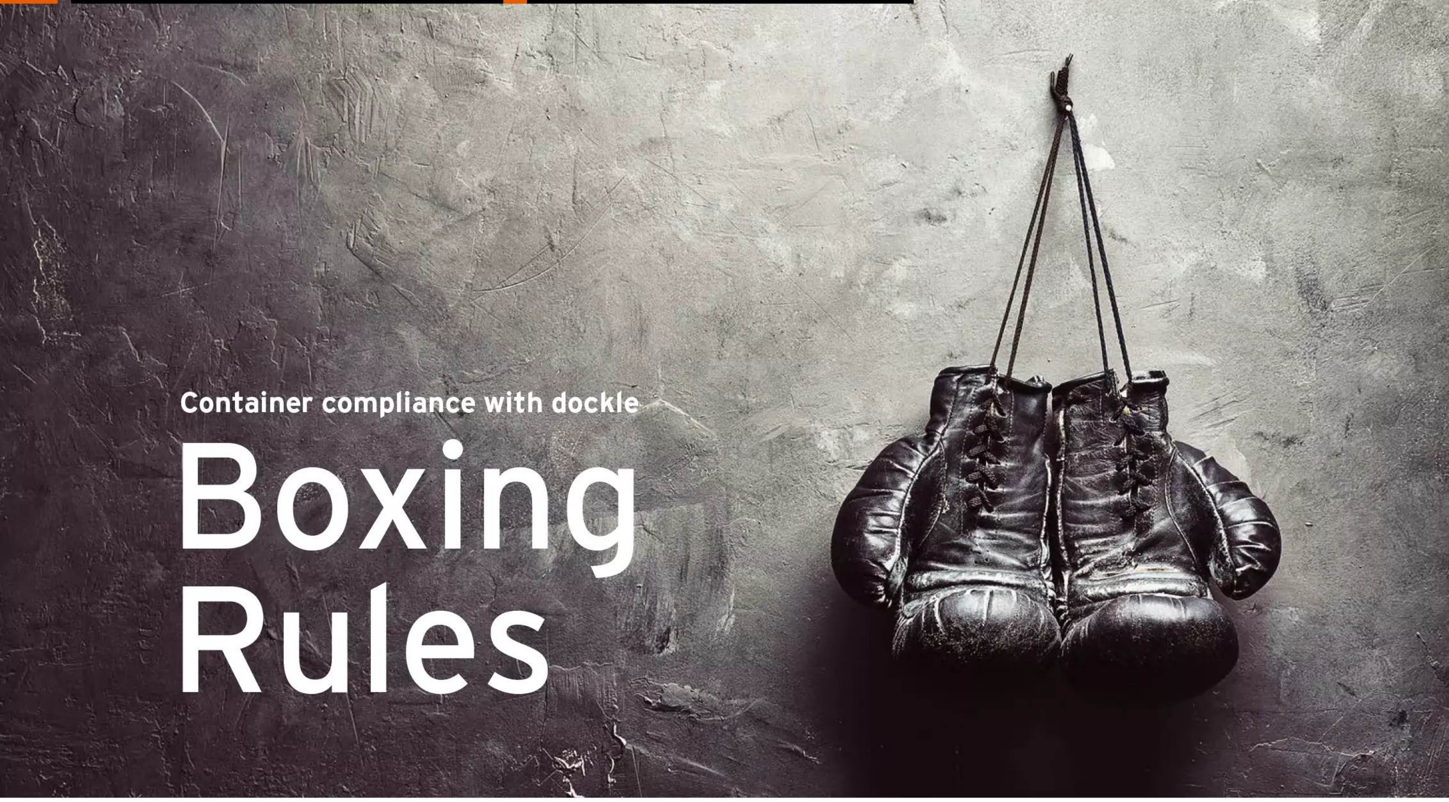
opensourcejobhub.com

@opensourcejobhub.com

Open Source JobHub

Open Source JobHub

@osjobhub@fosstodon.org



Container compliance with dockle

Boxing Rules

The open source dockle tool can help keep your Docker containers compliant. By Chris Binnie

If you work in cybersecurity, the mention of the word “compliance” invariably means that you are about to be subjected to a few hours of tedious, repetitive work. Ensuring that thousands of resources are compliant is no mean feat. These days, however, I definitely appreciate the need for systems and processes to meet external compliance frameworks and standards.

To-do lists can often help you stay organized, and this is where compliance tools come to the fore. Putting a check mark next to non-compliant resources after you have remediated them is really the only way to ensure consistent security across modern cloud or on-premise workloads and infrastructure.

In this article, I show you how to determine whether your Docker containers are compliant with the use of a fantastic open source tool called `dockle` [1]. The compliance standard that `dockle` checks against is the venerable CIS Benchmarks [2]. For good measure, `dockle` can also offer advice about Dockerfile linting (automatic checks of configuration files). If you need a little help constructing your Dockerfiles, you are gently reminded to look at the best practices page [3] on the Docker website.

On Your Marks

In this example, I look at installing the `dockle` binary file for Debian Linux derivatives (e.g., Ubuntu). As you can see in Listing 1, the `VERSION` variable is created to grab the latest version of the software from the `dockle/releases/latest` page in GitHub.

Listing 1: Installing `dockle`

```
VERSION=$( curl --silent "https://api.github.com/repos/goodwithtech/dockle/releases/latest" | grep
  '"tag_name":' | sed -E 's/.*"v([^"]+)".*/\1/' ) && curl -L -o dockle.deb https://github.com/
  goodwithtech/dockle/releases/download/v${VERSION}/dockle_${VERSION}_Linux-64bit.deb
$ sudo dpkg -i dockle.deb && rm dockle.deb
```

If I comment out the commands including and after the `&&`, the output offers the version *0.4.13* very neatly, without any other characters. The part I commented out uses this `VERSION` variable in the URL to download `dockle` with `curl`, and, on a separate line, install the `.deb` package file with the stalwart `dpkg` package manager.

I walked through the installation procedure in a bit more detail than usual, because other Linux distributions use very similar snippets for installing `dockle`. Have a look at the GitHub README [1] to confirm that even Windows (as incongruous as it seems) uses `curl` and the `VERSION` variable in the same way. However, it downloads a ZIP file containing an `.exe` file, as you would expect.

You might not be surprised in the slightest to learn that a Docker image also exists, which is how I will use `dockle`. Doing so also makes running `dockle` friendly to continuous integration and continuous deployment (CI/CD) pipelines; as you will see in a second, the `VERSION` variable line ensures always using the latest version.

Containers Scanning Containers

The Docker image approach from the GitHub README is shown in Listing 2. You can see that it uses a format similar to the Debian Linux derivative example in Listing 1. Obviously, you need to change `[YOUR_IMAGE_NAME]` for the container image you want to scan. To make any editing of this code snippet easier, you can paste it into a file

Lead Image © Viktor Gladkov, 123RF.com

Listing 2: `dockle` Container

```
$ VERSION=$( curl --silent "https://api.github.com/repos/goodwithtech/dockle/releases/latest" | grep
  '"tag_name":' | sed -E 's/.*"v([^"]+)".*/\1/' ) && docker run --rm -v /var/run/docker.sock:/var/run/
  docker.sock goodwithtech/dockle:v${VERSION} [YOUR_IMAGE_NAME]
```

Listing 3: `dockle_scan.sh`

```
#!/bin/bash

VERSION=$( curl --silent "https://api.github.com/repos/goodwithtech/dockle/releases/latest" | grep
  '"tag_name":' | sed -E 's/.*"v([^"]+)".*/\1/' ) && docker run --rm -v /var/run/docker.sock:/var/run/
  docker.sock goodwithtech/dockle:v${VERSION} $1
```

and make the file executable. Rather than declare an image name in the file, though, you can change `[YOUR_IMAGE_NAME]` to `$1`, so when you pass an image to your script file, you can name the image to scan. Listing 3 shows how I changed and saved the script to a file called `dockle_scan.sh`. Before scanning, pull down an NGINX container image from Docker Hub:

```
$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
1f7ce2fa46ab: Pull complete
[...snip...]
```

then scan the image, where the image name is passed to the script as `$1`:

```
$ ./dockle_scan.sh nginx:latest
Unable to find image ⮯
 'goodwithtech/dockle:v0.4.13' locally
v0.4.13: Pulling from goodwithtech/dockle
7b26c2f269ea: Pull complete
[...snip...]
```

The surprisingly detailed output from the `dockle` command is displayed in Figure 1. The `dockle` image is pulled down at the start of the process. In case you are wondering, Docker Engine will not repeatedly pull the latest version of the image; it's more efficient than that. The image ID is used to figure out whether it needs to download anything and will skip pulling the image again unless a new image ID doesn't match what is saved locally.

The output reveals a number of *setuid* files that will be executed as the owner of the file, irrespective of who runs them. When the owner is root, that can be bad news if the permissions are passed on unintentionally.

The output also mentions best practices around how you tag your images. For example, *nginx:latest* as an image name is too ambiguous for keeping track of versions and making sure you are secure with the most recent release. I've seen commit IDs from GitHub used as the container image tags, so there's no doubt who or what created that specific image. Also common is the use of in-house versioning as tags, but for personal projects, I tend to use a timestamp, because that pins the image to a specific second (e.g., *nginx:27169871000*), which is helpful if you create lots of images when testing.

Of more consequence to me, however, is the first warning line that mentions *Last user should not be root*. This warning from `dockle` is extremely important if you are running containers without any other well-considered security controls on the hosts that run your containers.

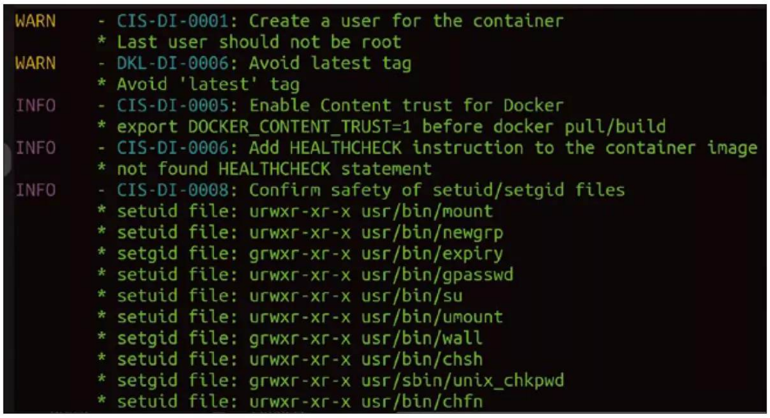
The problem arises when a Dockerfile is created, as shown here. The last line declares the user to be root:

```
USER: root
```

If you have created Dockerfiles in the past, you are most likely familiar with such an entry. Should that line exist, to all intents and purposes, the container is running as the root user when a container is spawned from the image. (For the purists, I'll cover the exception to that rule in a second.)

If you haven't guessed already, if the running container is compromised through many of the likely package vulnerabilities or non-compliance issues you've just seen `dockle` discover, then the underlying host is prone to offering the attacker root user permissions on a plate, which can mean taking over the whole host (and therefore all the containers on that host) or being able to move laterally between containers on the same host to gain access.

Whichever scenario an attacker succeeds in advancing, the results can offer a trove of treasures. The likelihood of then being able to get access to sensitive data through file shares (with full control of the host) and then potentially stealing secret keys that permit access to the cloud infrastructure as a whole, increases exponentially. Moving from one host to another is also much more likely. After all, hosts need access to other systems within on-premises or cloud infrastructure to operate properly. If the attacker becomes the root user, they get to see anything the host itself can see.



```
WARN    - CIS-DI-0001: Create a user for the container
        * Last user should not be root
WARN    - DKL-DI-0006: Avoid latest tag
        * Avoid 'latest' tag
INFO    - CIS-DI-0005: Enable Content trust for Docker
        * export DOCKER_CONTENT_TRUST=1 before docker pull/build
INFO    - CIS-DI-0006: Add HEALTHCHECK instruction to the container image
        * not found HEALTHCHECK statement
INFO    - CIS-DI-0008: Confirm safety of setuid/setgid files
        * setuid file: urwxr-xr-x usr/bin/mount
        * setuid file: urwxr-xr-x usr/bin/newgrp
        * setgid file: grwxr-xr-x usr/bin/expiry
        * setuid file: urwxr-xr-x usr/bin/gpasswd
        * setuid file: urwxr-xr-x usr/bin/su
        * setuid file: urwxr-xr-x usr/bin/umount
        * setgid file: grwxr-xr-x usr/bin/wall
        * setuid file: urwxr-xr-x usr/bin/chsh
        * setgid file: grwxr-xr-x usr/sbin/unix_chkpwd
        * setuid file: urwxr-xr-x usr/bin/chfn
```

Figure 1: Detailed output from an NGINX scan with **`dockle`**.

This well-known issue is definitely something to keep in mind. You should certainly get into the habit of ensuring that all of your images use a non-root user at the end of each Dockerfile. I mentioned the exception to the rule a second ago. If you think about how containers are run, you might be able to guess what I'm alluding to.

I have looked at the `USER` instruction in the Dockerfile that creates the image itself. Without any other intervention from the orchestrator (e.g., Kubernetes) or, in this case, the container engine (Docker Engine), the image will in most cases be run as the root user. However, you should also pay attention to the way the container is spawned from an image as it is deployed. Even if the image wants to run as root, it can be prevented from doing so and blocked from running at all. One mitigating control might include the use of user namespaces, where the user the system sees inside a running container doesn't directly correlate to the actual user (and UID and GID) on the underlying host.

In any case, it is better to construct Dockerfiles without the root user at the end. Simply create and use a non-root user on the host and in the Dockerfile for the image to use, then explicitly add that to the `USER` instruction every time. The workflow (Listing 4) means that a user is created on the host (I'll call it *nginx*, because I used that as an example image); then, early on in the Dockerfile, root user permissions are used to move more sensitive files and install packages that need root user permissions into the image. Once completed, the last instruction is the `USER` line that instructs the container engine to execute as the *nginx* user.

In hand with creating more sensible, secure images, you then also have security controls preventing developer mistakes (and commonly vendor mistakes, too!). The underlying hosts are protected from compromise through root-level permissions if an image tries to run as the root user.

The bare-bones version shown in Listing 4 required to fix the root user image issue is relatively straightforward when you have been shown what to do. This code should work (with a few tweaks) in most containers that run Debian Linux derivatives, such as Ubuntu. Make sure you understand the steps involved and fine-tune the Dockerfile to your requirements. The step I have missed before this is creating a user on the underlying host beforehand so the container user (in this case *nginx*) maps perfectly to the host's user.

The command to request a specific UID for a new user is

```
$ useradd -u 1001 nginx
```

which should also mean that the same GID is used by the new user as the declared UID. If you make a mistake, just run this command carefully:

```
$ userdel <username>
```

You can add the UID and GID to your Dockerfile as you like.

CI/CD Integration

Moving on, there's a number of pieces of functionality that I haven't covered with `dockle`, such as the CI/CD integration functionality. Listing 5 shows the example provided for a GitHub action failing on a `WARN` check level. (The `dockle` software comes with a variety of warnings (Table 1) that should help you grade the level of concern for a scanned image.) Dockle has further links in its README to example code snippets about using the tool in popular continuous integration services, including Travis CI, Circle CI, and GitLab CI.

Street Cred

If you integrate `dockle` with automated processes, you likely need a

Listing 4: Non-Root User Dockerfile

```
# Choose your poison (or more accurately, your Linux distro)
FROM debian

# Declare the USER, UID, and GID you want to use in the container
ARG USER=nginx
ARG UID=1001 # The UID of the "nginx" user you created on the host
ARG GID=$UID # Often the same as the UID

# Run some commands that need root user permissions (e.g., run a package update with "apt")
RUN apt update && apt install -y netcat

# Create your user (you may also want to "chown" files within and also declare a WORKINGDIR so that the
  user can read files in the directory you want the user to access after this)
RUN groupadd --gid $GID $USER && useradd --uid $UID --gid $GID -m $USER

# Finally, always execute as the non-root user, which is "nginx" in this case, and never the root user
USER $USER
```

Listing 5: Example GitHub Action

```
- uses: goodwithtech/dockle-action@main
  with:
    image: 'target'
    format: 'list'
    exit-code: '1'
    exit-level: 'warn'
    ignore: 'CIS-DI-0001,DKL-DI-0006'
```

Table 1: Warning Levels

Warning	Description
FATAL	An unwelcome warning; investigate further.
WARN	Ensure you are confident that the issue highlighted is either being mitigated to remove the risk or you fully understand the risks.
INFO	Informational, including potential performance hits, among other effects.
SKIP	Potentially, the target to be scanned isn't found.
PASS	No issues discovered.

way of exposing credentials so that `dockle` can read images from an image registry. The preferred way of doing so is to use the `export` command to declare environment variables.

To get you started, the `dockle` documentation offers code snippets for the Amazon Elastic Container Registry (AWS ECR), the Google Cloud Platform Container Registry (GCP GCR), and self-hosted registries (and how to toggle SSL on and off). In Listing 6 you can see example environment variables for adding Docker Hub credentials before running `dockle`.

Conclusion

Compliance doesn't have to be a word that fills you with dread. Instead, meeting compliance is really the only way to keep security consistent across multiple resources, especially when you are just dealing with your own resources and not thousands in an enterprise. Few standards are better than the CIS Benchmarks, which is backed by industry consensus.

As well as keeping you in line with compliance, `dockle` also provides useful linting advice to ensure that your Dockerfiles are efficient and sanely written. Also bear in mind the relative ease in which you can integrate compliance checks with CI/CD pipelines. You have little excuse to automate the compliance findings and then fail builds on any images that are out of step.

Finally, the relatively detailed insight into why running containers as root, and, more importantly, how to avoid doing so, should help in the future. Any images and running containers that you build and spawn as a result won't put your infrastructure at unnecessary risk. Stay vigilant. ■

Listing 6: Adding DockerHub Credentials

```
export DOCKLE_AUTH_URL=https://registry.hub.docker.com
export DOCKLE_USERNAME={DOCKERHUB_USERNAME}
export DOCKLE_PASSWORD={DOCKERHUB_PASSWORD}
```

Info

[1] dockle: [https://github.com/goodwithtech/dockle]

[2] CIS Benchmarks: [https://www.cisecurity.org/cis-benchmarks]

[3] Dockerfile best practices: [https://docs.docker.com/develop/develop-images/dockerfile_best-practices]

[4] Binnie, Chris. *Cloud Native Security.* Wiley, 2021: [https://www.amazon.co.uk/Cloud-Native-Security-Chris-Binnie/dp/1119782236]

Author

Chris Binnie is a Cloud Native Security consultant and co-author of the book *Cloud Native Security* [4].

shop.linuxnewmedia.com/shop

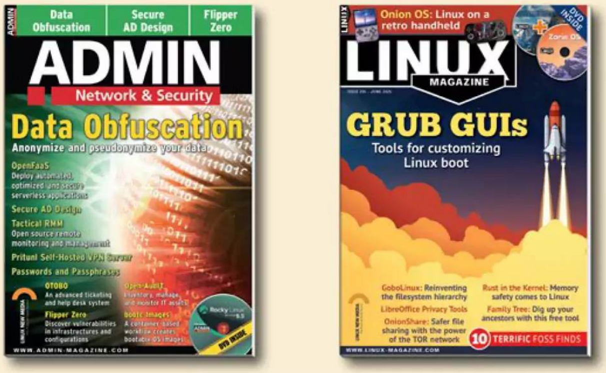
Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

Want to subscribe?

Searching for that back issue you really wish you'd picked up at the newsstand?

shop.linuxnewmedia.com/shop

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS

Configuring WireGuard with wg-portal

Instrument Panel

The wg-portal project helps you set up, manage, and scale a WireGuard VPN environment. By Matthias Wübbeling

WireGuard, the VPN server incorporated into the Linux kernel, has been available for some time and is becoming increasingly popular, replacing large VPN environments by traditional manufacturers on corporate networks. The main focus of WireGuard's development was to reduce the number of optional connection, encryption, and authentication parameters to the required minimum and to achieve a very small codebase as a result. What WireGuard lacks to support professional use is a management interface that handles the major share of the integration work for admins and clients.

Listing 1: `docker-compose.yaml`

```
services:
  wg-portal:
    image: wgportal/wg-portal:v2
    restart: unless-stopped
    cap_add:
      - NET_ADMIN
    network_mode: "host"
    ports:
      - "8888:8888"
    volumes:
      - /etc/wireguard:/etc/wireguard
      - ./data:/app/data
      - ./config:/app/config
```

The developers of the open source WireGuard Portal (`wg-portal`) [1] have set out to address exactly this problem. They are developing a web portal that supports admins, starting with creating the key material, through endpoint address management and routing configuration, to automatic access scheduling. Users can manage access independently through the self-service portal. In this article, I look at `wg-portal` installation and configuration and reveal the drawbacks involved in daily VPN use.

Docker Image

You need a WireGuard installation in place to try out the Docker image, although the installation does not have to be fully configured. The image provided by the developers [2] is useful for trying out `wg-portal` without too much overhead. Because you can modify the configuration by setting environment variables and because you need to integrate some volumes into the container, you will want to use the Docker `compose` function. The side effect is that it also makes it easier for you to try out different configurations. To get started, run the file shown in Listing 1.

At the time of writing, `wg-portal` had reached version 2. You might want to change the image label to the latest version, if the number has increased. The container requires `NET_ADMIN` authorizations to control the WireGuard network interfaces. If you now type

```
docker compose up
```

to launch the portal, you will notice that the image has not yet been preconfigured to the extent that you could simply start it out of the box. WireGuard Portal requires a configuration in `config/config.yml` at startup according to the log output. You might need root authorizations to create this file, because the `config` folder was created with root authorizations when the container was started for the first time. Add the following content to your `config.yml`:

```
core:
  admin_user:
    admin@admin-magazine.com
  admin_password: security-tip
  self_provisioning_allowed: true

web:
  external_url: http://localhost:8888
  request_logging: true
```

Lead Image © Pozitiv Studija; Fotolia.com

The self-provisioning line lets users create their own VPN connections, in contrast to the default configuration. After restarting the container, the message about a missing configuration no longer appears, and you can open *http://localhost:8888* in your browser to access the web interface, which takes you to the login page or an overview page with general information about WireGuard and `wg-portal` (select *Login*). Now enter the access data from the `config.yml` file in the login box and click *Sign In*.

After logging in, open your profile, where you can manage the VPN profiles for your current user account. Of course, you will not want to save any information there at this point. When you press the button to add a connection, you are taken to an input dialog where you can configure your connection with the parameters that you are already familiar with from the classic WireGuard configuration.

Managing Interfaces

Selecting the *Interfaces* item in the top bar takes you to the overview of existing WireGuard endpoints; `wg-portal` also recognizes and lists endpoints that you created manually. If you select the small eye symbol in the titlebar, `wg-portal` shows the text configuration for the interface. You can also download this configuration directly or edit it in a dialog after clicking on the settings icon. If you have not yet configured an interface or would like to create another, simply press the + button to create the configuration in the dialog.

The current endpoints are the connections that exist for the endpoints and their current statuses. You can also change these configurations. When it displays the connection configuration, `wg-portal` also displays a QR code that can be used on a smartphone with WireGuard as a convenient way of transferring the configuration to the device. Of course, this only works if the private key for the connection is also stored in `wg-portal`; otherwise, you need to add it on the device later.

User Management and SSO

After familiarizing yourself with the interface and connection configurations, you can start to manage users under the *Users* item in the menubar; `wg-portal` has its own user management feature. To add new users, press the + button in the overview. In contrast to admin users, normal users can only view their own profiles and, if they modified their configurations, also configure their own connections.

Besides local users, `wg-portal` also lets single sign-on (SSO) services connect with, for example, OpenID Connect (OIDC), OAuth2, or LDAP, which allows you to integrate WireGuard and `wg-portal` seamlessly with your existing infrastructure in terms of user management. To connect an LDAP server, simply extend the configuration as shown in Listing 2.

Adjust the settings to match your infrastructure and then restart the container. In the user overview, you will now see the users from your LDAP in the Source column. Connecting to OIDC or OAuth2 is equally easy, and you can even configure the use of multiple authentication back ends.

Additional Manual Work

Even if `wg-portal` does some of the work for you, such as managing the interfaces and the automatic or self-service-oriented user administration for your WireGuard VPN, you still have some work to do; for example, `wg-portal` does not configure the firewall, although it is important to isolate users on different subnets or only allow selected users to access certain network resources.

In smaller setups, where you might want to hide users automatically behind network address translation (NAT), you can implement these rules with Bash commands in what are known as "interface hooks." These hooks can be configured for each individual connection; they are not executed server-side, but on the clients, and you will always have to configure and maintain complex networks as a background task.

Listing 2: Extending `config.yml`

```
[...]
auth:
  ldap:
    - id: LDAP01
    url: ldap://ldap.admin-magazine.com
    start_tls: true
    base_dn: "DC=admin-magazine, DC=com"
    bind_user: wgportal
    bind_pass: secret
    admin_group: dc=WG-ADMINS
    synchronize: true
```

Conclusions

WireGuard is gaining ground as a VPN environment. For professional use in the corporate environment, you will probably still miss the kind of toolkit that takes most of the work off your hands. The `wg-portal` project steps in with support for setting up and managing WireGuard, and it offers self-service for users, as well, which is an easy way to scale irrespective of the number of users. ■

Info

[1] WireGuard Portal: [https://wgportal.org/latest/]

[2] wg-portal Docker image: [https://hub.docker.com/r/wgportal/wg-portal]

The Author

Dr. Matthias Wübbeling is an IT security enthusiast, scientist, author, consultant, and speaker. As a Lecturer at the University of Bonn in Germany and Researcher at Fraunhofer FKIE, he works on projects in network security, IT security awareness, and protection against account takeover and identity theft. He is the CEO of the university spin-off Identeco, which keeps a leaked identity database to protect employee and customer accounts against identity fraud. As a practitioner, he supports the German Informatics Society (GI), administrating computer systems and service back ends. He has published more than 100 articles on IT security and administration.

Visualizing IT infrastructures with IVRE

Mapped

The IVRE tool integrates into existing IT infrastructures for reconnaissance on private networks to help analyze endpoints accessible over the Internet.

By Matthias Wübbeling

Shodan [1] is probably the best-known tool for assessing the threat situation for your organization's network. Of course, Shodan can only check your network from the outside. Various tools such as Nmap, Masscan, or ZGrab can gain similar insights into the network behind the demilitarized zone (DMZ). The open source IVRE [2] tool provides a framework containing all the components you need to obtain results in a format that is just as clear and familiar as those from Shodan.

Although IVRE lets you scan and map your IT infrastructure, it can also analyze the results collectively and display them for interaction. IVRE also imports the output of common scanning tools and merges it into its own database, which means you can integrate IVRE with your existing toolset. In this article, I discuss the feature set that IVRE provides and show you how to set up IVRE for testing and to import initial data into the framework.

Installing and Launching

IVRE has some dependencies and is not always as easy to install as the website claims. If you use Kali Linux in your environment, you can use `apt` to install on Kali without problems, in most cases. In my setup, I launched IVRE on Linux and used the Vagrant description provided by the developers on GitHub [3] by cloning the repository with the command:

```
git clone https://github.com/ivre/ivre
```

Next, create the directories for IVRE's work data and set the access rights accordingly:

```
mkdir -m 777 var_lib_mongodb ➘
  ivre-share dokuwiki_data
```

The Vagrant description relies on a Docker back end and caused issues from time to time when I was downloading images while writing this article. If `vagrant up` does not work on the first attempt, you can try running the command a couple more times. As an alternative, you could also adapt the Docker definition provided in the `docker-compose.yaml` file and use it directly. In both cases, five Docker containers are used: one container each for the client application, for MongoDB as the back-end database, for the WSGI server for the API, for the web server as the graphical interface, and finally for DokuWiki to document the results.

If the launch procedure worked, you can now type *http://localhost/* in your browser's address bar to access the IVRE web application. At this point you will not see any content because IVRE has not yet transferred the results data to the database. You now have a number of options for generating the data to import. Theoretically, you could now simply start Nmap on your system (e.g., to scan a private home network); just swap in the correct network prefix in the command:

```
nmap -Sv -oX localnet.xml 192.168.0.0/24
```

The results of the scan are stored in the `localnet.xml` file. The next step is to publish the file in the container for processing by IVRE downstream. In

Lead Image © Victoria, Fotolia.com

this trial environment, calling Nmap directly from inside the container with IVRE is easier. To start, run a command to discover the ID or name of the Docker container and launch an interactive shell in the container:

```
docker ps
docker exec -ti ivreclient /bin/bash
```

You can now work directly with IVRE at the command line. If you use the same environment as before, the command is:

```
ivre runscans --network 192.168.0.0/24 ➘
              --output=XMLFork
```

Now you need to give IVRE some time to scan your network, unfortunately without the help of a progress bar. If you return to the prompt and do not see any serious errors in the output, you can now run the following command to transfer the results of the XML files from all accessible computers to the IVRE database:

```
ivre scan2db -c Admin-Magazine -s NMAP ➘
   -r scans/NET-192.168.0.0-24/up
```

The `-c` switch lets you enter an arbitrary name for the entry category, which could also be a reference to network segments or the tools you used. The output from the import shows the number of results. If you check the web interface again directly after importing, you might be surprised to see nothing. The data is now in the database and can be searched by typing at the prompt,

```
ivre scancli --port 80
```

which reports the Nmap results for all endpoints that use port 80.
To make the data available in the web interface, run

```
ivre db2view
```

The output also indicates that IVRE is trying to collect additional information on the endpoints found from other sources. To use this information, run the

```
ivre getwebdata
```

command and then restart provisioning with `db2view`.
You have now successfully prepared the results for display with only a few minor warnings. You need to run this command regularly to make sure you have up-to-date data from these external sources at all times. If you refresh the web view again now, you should see some initial results on your dashboard.
IVRE has options for filtering and searching against the data with various criteria. For example, you could search for specific IP ranges, ports, or services to select specific endpoints and information. This filtering helps focus attention on the most important results and identify potential vulnerabilities more quickly.
You can add an mnemonic entry to the integrated DokuWiki for each endpoint the program finds. To do so, click on the small pencil icon at the edge of the results map to open DokuWiki in the right-hand window area, where you can create the documentation.

Active vs. Passive Monitoring

What you have seen so far with IVRE has all been based on manual (active) execution, but you can continue to provide data passively in the background, which IVRE then analyzes and processes. Active monitoring is about scanning individual networks and collecting data related to endpoints, open ports, running services, and other components they contain. This method is particularly effective if you are looking for an ad hoc overview of the structure and security of a network.
A typical example of an active scan is the use of Nmap, as mentioned before. By integrating the Nmap scans into IVRE, the data can be analyzed in detail and visualized. However, active monitoring is not without risks – especially in production environments where it can be conspicuous and potentially trigger alarms.
In contrast, passive monitoring (i.e., a more traditional approach in which no direct interaction occurs with the target systems) instead uses information collected from publicly accessible sources, such as network traffic monitoring, DNS queries, or data from public services. IVRE supports passive monitoring by analyzing data that can be permanently imported from various sources in the background. You could also use this passive approach to implement offline monitoring (e.g., with packet capture, PCAP, files created from network data acquisition) to obtain interesting information about the monitored network in a less invasive way.

Conclusions

IVRE is more than just a tool for reconnaissance on your private network. Its flexible framework can be integrated easily into existing IT infrastructures, supporting both active and passive network detection use cases. The ability to integrate various tools, the automation options, and the visualization tools make IVRE a great local alternative to commercial services such as Shodan. Once set up in a production environment, IVRE helps you secure your network and detect threats at an early stage. ■

Info

[1] Shodan: [https://www.shodan.io]
[2] IVRE: [https://ivre.rocks]
[3] IVRE on GitHub: [https://github.com/ivre/ivre]

The Author

Dr. Matthias Wübbeling is an IT security enthusiast, scientist, author, consultant, and speaker. As a Lecturer at the University of Bonn in Germany and Researcher at Fraunhofer FKIE, he works on projects in network security, IT security awareness, and protection against account takeover and identity theft. He is the CEO of the university spin-off Identeco, which keeps a leaked identity database to protect employee and customer accounts against identity fraud. As a practitioner, he supports the German Informatics Society (GI), administrating computer systems and service back ends. He has published more than 100 articles on IT security and administration.

Identifying and using software licenses

Small Print

Software licenses tell you how you can use software packages and legally deploy the software in your projects. Fortunately, tools help you automate the process of verifying licenses. By Frank Hofmann

If you derive a program component from a GNU General Public License (GPL) software package, you are obligated to publish your software under the same GPL license. However, if the program is a combination of GNU Lesser General Public License (LGPL) code and your own work, you are allowed to distribute the result under a new, potentially proprietary license. You would have the same options if you derived your software from a Berkeley Software Distribution (BSD)-licensed package.

Although GPL, LGPL, and BSD are all free software licenses, they envisage very different types of use. To illustrate the fundamental differences in more detail, software licenses can be broken down into different categories.

Outside of the open source world, proprietary software licenses are the norm, and they rarely follow a standard. Besides the widespread commercial licenses, this category also includes shareware and freeware.

Categorical

Free and open source software licenses are defined by the Free Software Foundation (FSF) [1] and the Open Source Initiative (OSI) [2]. A distinction can essentially be made between copyleft, permissive, and public domain licenses.

Copyleft licenses [3] oblige the licensee to publish any adaptations of the software (derivatives) under the same license as the original work.

This type of license, also known as a reciprocal license, is intended to prevent restrictions on the use of the software. The best-known copyleft license is undoubtedly the GPL; its copyleft is considered to be very strict, whereas that of the Mozilla Public License (MPL) is viewed as quite permissive. Because the licensors themselves are not bound by their own copyleft, they can publish new versions under a proprietary license or allow third parties to do the same, which is referred to as multiple licensing.

Copyleft licenses can quickly lead to incompatibilities if software is distributed along with software subject to other free licenses. For example, the three-clause BSD license is incompatible with the GPL, whereas the European Union Public License (EUPL) [4] is an interoperable reciprocal license that is compatible with most other open reciprocal licenses. The compatible license obligations take precedence if they conflict with the obligations arising from the EUPL [5].

Permissive open source licenses allow broader reuse than copyleft licenses. Derivations and copies of the source code can be distributed under conditions that have fundamentally different characteristics than the original license. The best-known examples of permissive open source licenses are the MIT and BSD licenses.

Finally, the public domain licenses transfer the copyright to the general public. The Unlicense [6] was created to mark software as public domain.

Beyond Software

Some open source software licenses can also be used for non-software works. In fact, they are often the best choice, especially if the works in question are edited as source code and versioned.

In contrast, the popular Creative Commons licenses are explicitly not intended for software [7], but for data, media, and documentation. The Open Knowledge Foundation [8] formulated the Open Data Commons licenses [9] for data and databases. Table 1 compares the non-software license variants, which now also include those for open source hardware [10].

As mentioned earlier, you are only free to choose the license for your programs if you have not derived your work from other software or combined it with other software. License compatibility for derived or combined works made up of your own software and external code that is subject to an open source license is shown in Figure 1.

Lead Image © alphaspirit, 123RF.com

Table 1: Licenses Beyond Software

License	Comment
Data, media, content	
Creative Commons	Open licenses for non-software material, from datasets to videos; CC licenses are not recommended for software
Open Data Commons	Licenses for data (databases) of the Open Knowledge Foundation
DL-DE->BY-2.0 [35] DL-DE->Zero-2.0 [36]	The German Open Government data portal has published two Data License Germany versions
Community Data License Agreements (CDLAs) [37]	Linux Foundation data licenses
Free Art License [38]	License for artistic works
Documentation	
GNU Free Documentation License (FDL) [39]	Copyleft license for documentation; intended for use with all GNU manuals; its applicability is limited to text-based works such as books
FreeBSD Documentation License [40]	Permissive documentation license with copyleft that can be combined with the GNU FDL
Open Publication License 1.0 [41]	Free documentation license with copyleft, provided none of the license options from section VI of the license are exercised; this license cannot be combined with the GNU FDL
Fonts	
SIL Open Font License 1.1 [42]	License for fonts that may be used freely in other works
GNU General Public License 3 (GPLv3) [43]	The GPLv3 can also be used for fonts, but they may only be included in documents with the font exception
LaTeX EC Fonts [44]	Free fonts (European Computer Modern, Text Companion), which are usually used with LaTeX
Arphic Public License [45]	Free license with copyleft
IPA Font License [46]	Free license with copyleft, but whose derived values may not use or contain the name of the original
Hardware (CERN Licenses)	
CERN-OHL-P-2.0 CERN-OHL-W-2.0 CERN-OHL-S-2.0 [47]	CERN Open Hardware Licence (OHL) v2 permissive CERN Open Hardware Licence (OHL) v2 weakly reciprocal CERN Open Hardware Licence (OHL) v2 strongly reciprocal

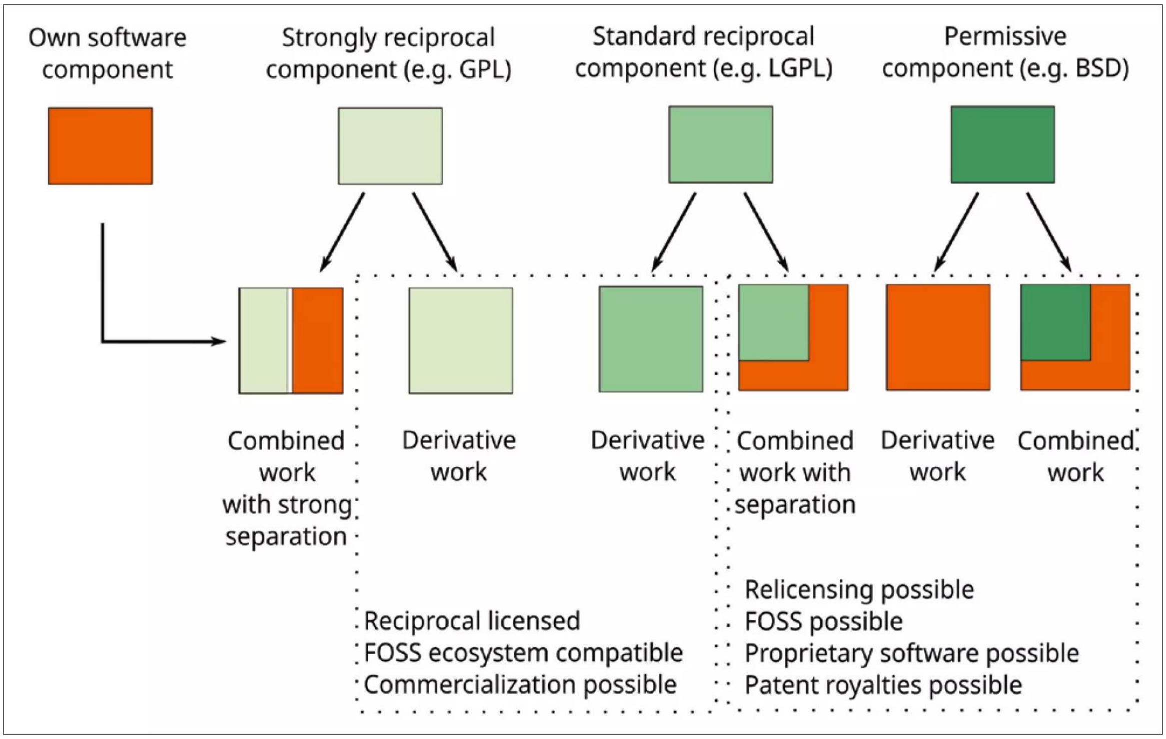

Own software component
Strongly reciprocal component (e.g. GPL)
Standard reciprocal component (e.g. LGPL)
Permissive component (e.g. BSD)
Combined work with strong separation
Derivative work
Derivative work
Combined work with separation
Derivative work
Combined work
Reciprocal licensed
FOSS ecosystem compatible
Commercialization possible
Relicensing possible
FOSS possible
Proprietary software possible
Patent royalties possible

Figure 1: Overview of the compatibility of derived software licenses [12]. CC0 "No Rights Reserved" [11]

Various methods, including manual and software-based approaches, can be used to discover the license for derived or combined works. The manual method applies if the project is hosted on GitHub: The license is either stated in the `license` file or can be found in the `license/` directory. To discover more, check the project repository to see if you can find a license there. If not, the next step is to look at the notes in the `README` file. Licenses can also be identified by software tools such as Liccheck [13] and REUSE [14].

Listing 1: License Check with Liccheck

```
$ liccheck -s liccheck.ini -r requirements.txt
gathering licenses...
3 packages and dependencies.
check unknown packages...
3 packages.
  cffi (1.15.1): ['MIT']
    dependency:
      cffi << cryptography
  cryptography (41.0.3): ['Apache Software', 'BSD']
    dependency:
      cryptography
  pycparser (2.21): ['BSD']
    dependency:
      pycparser << cffi << cryptography
```

Listing 2: Reuse Test Results

```
$ cd cryptography
$ reuse lint

# MISSING COPYRIGHT AND LICENSE INFORMATION
The following files have no copyright and license
information:
* .gitattributes
* .github/ISSUE_TEMPLATE/openssl-release.md
[...]
* vectors/cryptography_vectors/x509/wosign-bc-invalid.pem
* vectors/pyproject.toml
The following files have no license information:
* docs/_ext/linkcode_res.py
* src/cryptography/__about__.py

# SUMMARY
* Incorrect licenses: 0
* Outdated licenses: 0
* Licenses without file extension: 0
* Missing licenses: 0
* Unused licenses: 0
* Used licenses: 0
* Read errors: 0
* files with copyright information: 2 / 2806
* files with license information: 0 / 2806

Unfortunately, your project is not compliant with
   version 3.0 of the REUSE specification :-( $
```

Liccheck

Liccheck is interesting for anyone who uses Python in their projects – either directly or as a package dependency. One of the fixed components of every Python package is a file named `requirements.txt`, where the programmer specifies which other Python packages the current package needs to run. Liccheck looks for licenses of the packages listed in this file and displays any license conflicts it detects.

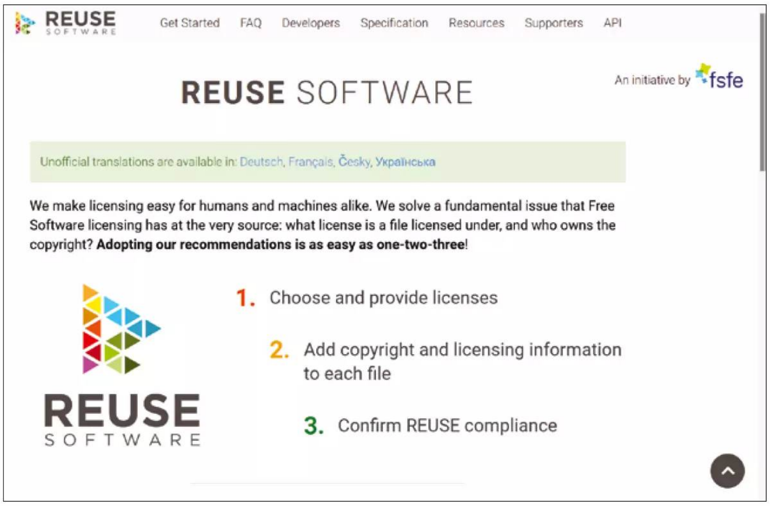
You first need to define a strategy that Liccheck will use to carry out its check; that is, which licenses should Liccheck consider to be in violation and which not? The more precise this definition, the more accurate the results of the check. Listing 1 shows how Liccheck is called and the results of the analysis for the Python *cryptography* module [15].

REUSE

The REUSE project [14] was created under the auspices of the Free Software Foundation Europe (FSFE) and is based on two pillars: machine-readable copyright files from the Debian project and Software Package Data Exchange (SPDX) [16], a standardized method for exchanging copyright and license information between projects and people [17].

Table 2: SPDX Identifiers (Selection)

Official License Name	SPDX Identifier
GNU Affero General Public License v3.0 or later	AGPL-3.0-or-later
Apache License 2.0	Apache-2.0
BSD 3-Clause "New" or "Revised" License	BSD-3-Clause
Common Development and Distribution License 1.1	CDDL-1.1
Creative Commons Attribution Share Alike 4.0 International	CC-BY-SA-4.0
Deutsche Freie Software Lizenz (German Free Software License)	D-FSL-1.0
European Union Public License 1.2	EUPL-1.2
GNU Free Documentation License v1.3 or later	GFDL-1.3-or-later
GNU General Public License v3.0 or later	GPL-3.0-or-later
GNU Lesser General Public License v3.0 or later	LGPL-3.0-or-later
Mozilla Public License 2.0	MPL-2.0
XFree86 License 1.1	XFree86-1.1



REUSE SOFTWARE

Unofficial translations are available in: Deutsch, Français, Česky, Українська

We make licensing easy for humans and machines alike. We solve a fundamental issue that Free Software licensing has at the very source: what license is a file licensed under, and who owns the copyright? **Adopting our recommendations is as easy as one-two-three**!

1. Choose and provide licenses
2. Add copyright and licensing information to each file
3. Confirm REUSE compliance

Figure 2: The homepage of the REUSE project. © FSFE e.V.

Listing 3: Multiple Licenses

```
This software is made available under the terms of *either* of the
licenses found in LICENSE.APACHE or LICENSE.BSD. Contributions to
cryptography are made under the terms of *both* licenses.

The code used in OpenSSL locking callback and OS random engine is
derived from the same in CPython, and is licensed under the terms
of the PSF License Agreement.
```

REUSE provides both instructions and the `reuse` tool [18] to help you ensure that your project has correct licensing information. The project website (Figure 2) compares REUSE with other solutions [19], including ScanCode [20], ClearlyDefined [21], FOSSology [22], and OpenChain [23].

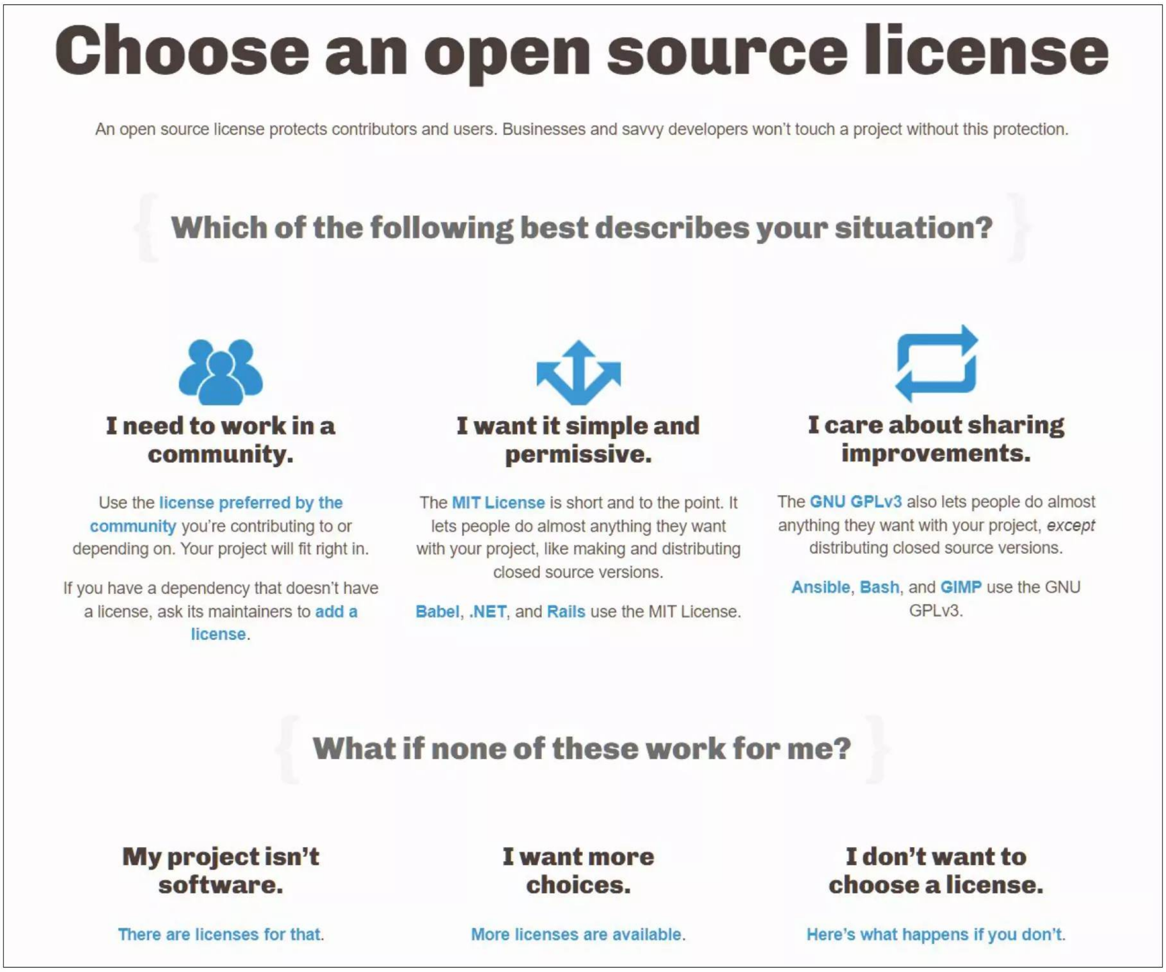
The installation procedure with `pip` for the Python `reuse` command-line tool is fully documented. The software was added to the package sources as a Debian package [24] under Debian GNU/Linux 12 "bookworm."

The `reuse` tool offers a whole range of useful functions, which are certainly not limited to adding copyright and license information in the header of a file (`annotate` subcommand) or preparing a project to comply with the REUSE specifications (`init`). You can also check a project for compliance with the REUSE specifications (`lint`) and create an SPDX document with all the files from the project (`spdx`). Listing 2 shows the `reuse check` results for the Python *cryptography* module. Even at first glance, it is clear that several files do not contain copyright and licensing information and that the module therefore does not comply with the REUSE specification.

Selecting a License

Once the licenses for the individual works have been clarified, it is time to choose the license for your own project. Both GitHub and SPDX offer assistance. To begin, select the appropriate SPDX terms [25] and then add them to the license description header or your software. Table 2 lists the most important SPDX identifiers.

After selecting a license, you need to compare it with the licenses of the works you incorporated, check for conflicts, and ensure you are in compliance with the desired standard by trying `reuse` again. If no conflicts are



Choose an open source license

An open source license protects contributors and users. Businesses and savvy developers won't touch a project without this protection.

Which of the following best describes your situation?

I need to work in a community.

Use the license preferred by the community you're contributing to or depending on. Your project will fit right in.

If you have a dependency that doesn't have a license, ask its maintainers to add a license.

I want it simple and permissive.

The MIT License is short and to the point. It lets people do almost anything they want with your project, like making and distributing closed source versions.

Babel, .NET, and Rails use the MIT License.

I care about sharing improvements.

The GNU GPLv3 also lets people do almost anything they want with your project, *except* distributing closed source versions.

Ansible, Bash, and GIMP use the GNU GPLv3.

What if none of these work for me?

My project isn't software.

There are licenses for that.

I want more choices.

More licenses are available.

I don't want to choose a license.

Here's what happens if you don't.

Figure 3: Online guides help you choose the right license for your software project. © GitHub Inc.

Creating a License File on GitHub

GitHub can help you create an open source license for a repository in a number of steps (see also the "Including License Logos" box). First, go to the main page of your repository, select *Create new file*, and enter *LICENSE* or *LICENSE.md* as the file name. Next, click on the *Choose a license template* button to the right of the file name field.

A selection field for the appropriate open source license opens. If the selected license requires additional information, you will be presented a prompt for that purpose. After specifying a commit message such as *Add license*, a click on *Commit new file* will complete the process.

If a `LICENSE` file already exists in the repository, GitHub uses the Licensee Ruby Gem [48] to match it with a short list of open source licenses. If GitHub cannot recognize the license in the repository, it might contain multiple licenses or be too complex. You then need to check whether you can simplify the licensing structure (e.g., by outsourcing the complexity to the `README` file).

Conversely, GitHub offers to search for repositories with specific licenses or license families. An overview of the keywords used by GitHub relating to licenses can be found in *Searching GitHub by license type*.

found, everything is fine. If the check does reveal a conflict, you need to reconsider your choice of license and repeat the check step.

Never Without a License

A software license is simply part of the software. For free software, you can choose from a very wide selection of licenses [26]. OSI, for example, provides an overview [2].

As the author, you choose appropriate license terms that you attach to your work in text form. The license is your way of defining what other people can do with your work in both private and commercial use. Multiple licensing exists if you specify different licenses for the two types of use, as shown in Listing 3 for the Python *cryptography* module.

Some traps can later lead to unwanted conflicts (e.g., if you use the Server Side Public License, SSPL [27]). OSI has decided that the license is not consistent with its open source definition. However, it is used in major software projects such as MongoDB, the Elasticsearch search engine, and the Kibana visualization tool.

You can choose a proprietary software license, but with no standards, comparisons are more complicated. Programs licensed in this way can be commercial in the narrower sense, but also shareware or freeware. In the case of public domain licenses, the copyrights are transferred to the general public. The Creative Commons Zero License [11] was created to mark the release of the broadest possible rights of use.

On the other hand, if you do not attach a license to your work at all, any interested party would first have to ask you what kind of use you intend for your software (i.e., what rights you grant and what obligations a user has). The same applies vice versa if you want to use software or libraries to which the author has not attached a license. The onus of clarifying the situation lies with the user.

Although this query method is the standard procedure if you use other people's work, it is a pain and prevents quick decisions often needed nowadays. You have no way to rule out the interested party either using your work without asking for your permission first or choosing some other clearly and unambiguously licensed work instead to avoid a potential dispute. Harald Welte has been documenting this kind of dispute since 2004 in the scope of his GPL Violations [28] project.

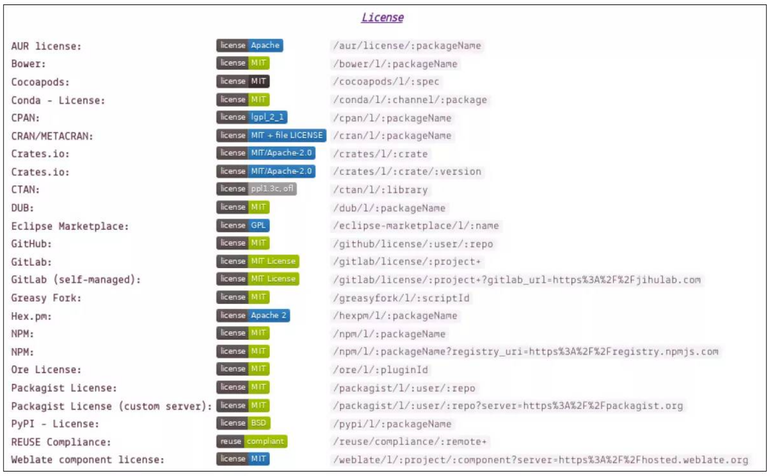
Assistance

Choosing the right license is not always easy, which has prompted several projects and has led to the Free Software Foundation [29], GitHub [30], and Wikipedia [31] going to the trouble of offering online guides (Figure 3). They compare the objectives of the various free licenses [32] and describe their use cases [33]. The FSFE also offers its support in verifying license compliance in the form of legal advice from a legal expert [34].

These options help you keep track and make it easier to find your way through the legal jungle when it comes to individual projects. In the case of multiple licensing, as shown in the example in Listing 3, or of different licenses for the files and components, GitHub will not work. It cannot handle this kind of case cleanly (i.e., thus far you can only

Including License Logos

To make it easier for potential users to recognize the license under which you publish a work, you can add a catchy license shield (badge). The Shields.io [49] project takes this step off your hands and offers both ready-made and individually generated license shields (Figure 4), which you then only need to add to the documentation for your project.



License

AUR license:	license Apache	/aur/license/:packageName
Bower:	license MIT	/bower/l/:packageName
Cocoapods:	license MIT	/cocoapods/l/:spec
Conda - License:	license MIT	/conda/l/:channel/:package
CPAN:	license lgpl_2_1	/cpan/l/:packageName
CRAN/METACRAN:	license MIT + file LICENSE	/cran/l/:packageName
Crates.io:	license MIT/Apache-2.0	/crates/l/:crate
Crates.io:	license MIT/Apache-2.0	/crates/l/:crate/:version
CTAN:	license ppl1.3c, ofl	/ctan/l/:library
DUB:	license MIT	/dub/l/:packageName
Eclipse Marketplace:	license GPL	/eclipse-marketplace/l/:name
GitHub:	license MIT	/github/license/:user/:repo
GitLab:	license MIT License	/gitlab/license/:project+
GitLab (self-managed):	license MIT License	/gitlab/license/:project+?gitlab_url=https%3A%2F%2Fjihulab.com
Greasy Fork:	license MIT	/greasyfork/l/:scriptId
Hex.pm:	license Apache 2	/hexpm/l/:packageName
NPM:	license MIT	/npm/l/:packageName
NPM:	license MIT	/npm/l/:packageName?registry_uri=https%3A%2F%2Fregistry.npmjs.com
Ore License:	license MIT	/ore/l/:pluginId
Packagist License:	license MIT	/packagist/l/:user/:repo
Packagist License (custom server):	license MIT	/packagist/l/:user/:repo?server=https%3A%2F%2Fpackagist.org
PyPI - License:	license BSD	/pypi/l/:packageName
REUSE Compliance:	reuse compliant	/reuse/compliance/:remote+
Weblate component license:	license MIT	/weblate/l/:project/:component?server=https%3A%2F%2Fhosted.weblate.org

Figure 4: Some examples of the license shields generated by Shields.io. © Shields.io

store single-license projects there; see the “Creating a License File on GitHub” box).

Conclusions

In this article, I looked at the various software licenses, discussed their use, and pointed out the pitfalls that need to be taken into account. I also introduced you to the methods and tools you can use to identify the license of any software you use. In a follow-up article, I will discuss how to add one or more licenses to your own Python package and document it. I look at procedures and tools that have proven their value in everyday life and show you how to integrate them into your development workflow. ■

Info

[1] Free Software Foundation: [https://www.fsf.org/]

[2] OSI approved licenses: [https://opensource.org/licenses]

[3] Software licenses: [https://en.wikipedia.org/wiki/Software_license#Free_and_open-source_software_licenses]

[4] European Union Public License: [https://joinup.ec.europa.eu/collection/eupl/introduction-eupl-licence]

[5] “Matrix of EUPL compatible open source licences”: [https://joinup.ec.europa.eu/collection/eupl/matrix-eupl-compatible-open-source-licences]

[6] Unlicense: [https://choosealicense.com/licenses/unlicense/]

[7] The Creative Commons license and software: [https://creativecommons.org/faq/#can-i-apply-a-creative-commons-license-to-software]

[8] Open Knowledge Foundation: [https://okfn.org]

[9] Open Data Commons: [https://opendatacommons.org]

[10] Open Source Hardware Association: [https://oshwa.org]

[11] CC0: [https://creativecommons.org/share-your-work/public-domain/cc0/]

[12] Välimäki, Mikko. 2005. *The Rise of Open Source Licensing: A Challenge to the Use of Intellectual Property in the Software Industry*, PhD dissertation, Helsinki University of Technology, Helsinki, Finland, May 2005, pg. 119. Turre Publishing, 2005: [https://commons.wikimedia.org/w/index.php?curid=45969489]

[13] Liccheck: [https://github.com/dhatim/python-license-check]

[14] REUSE: [https://reuse.software]

[15] Python cryptography module: [https://github.com/pyca/cryptography]

[16] SPDX: [https://spdx.dev]

[17] License information exchange: [https://jupyter-tutorial.readthedocs.io/en/latest/dashboards/index.html]

[18] reuse tool: [https://github.com/fsfe/reuse-tool]

[19] Comparing reuse with other solutions: [https://reuse.software/comparison/]

[20] ScanCode: [https://github.com/nexB/scancode-toolkit]

[21] ClearlyDefined: [https://clearlydefined.io]

[22] FOSSology: [https://www.fossology.org]

[23] OpenChain: [https://www.openchainproject.org]

[24] Debian reuse package: [https://packages.debian.org/bookworm/reuse]

[25] SPDX license list: [https://spdx.org/licenses/]

[26] “Free Software, Open Source, FOSS, FLOSS – same but different” by Björn Schießle, FSFE, December 8, 2012: [https://fsfe.org/freesoftware/comparison.en.html]

[27] Server Side Public License: [https://en.wikipedia.org/wiki/Server_Side_Public_License]

[28] License disputes: [https://gpl-violations.org]

[29] “How to Choose a License for Your Own Work”: [https://www.gnu.org/licenses/license-recommendations.html]

[30] Choose a License: [https://choosealicense.com]

[31] “Comparison of free and open-source software licenses”: [https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses]

[32] License overview: [https://www.gnu.org/licenses]

[33] License comparison: [https://choosealicense.com/appendix/]

[34] Checking for license compliance: [https://mastodon.social/@fsfe/110723128814901660]

[35] DL-DE->BY-2.0: [https://www.govdata.de/dl-de/by-2-0] (in German)

[36] DL-DE->Zero-2.0: [https://www.govdata.de/dl-de/zero-2-0] (in German)

[37] Community Data License Agreement: [https://cdla.dev]

[38] Free Art License: [https://artlibre.org/licence/lal/en/]

[39] GNU Free Documentation License: [https://www.gnu.org/licenses/fdl-1.3.txt]

[40] FreeBSD Documentation License: [https://www.freebsd.org/copyright/freebsd-doc-license/]

[41] Open Publication License: [https://opencontent.org/openpub/]

[42] SIL Open Font License: [https://opensource.org/licenses/OFL-1.1]

[43] GNU General Public License 3.0: [https://www.gnu.org/licenses/gpl-3.0]

[44] License for EC fonts: [https://mirrors.ctan.org/fonts/ec/src/copyrite.txt]

[45] Arphic Public License: [https://spdx.org/licenses/Arphic-1999]

[46] IPA Font License: [https://spdx.org/licenses/IPA.html]

[47] CERN Open Hardware Licence version 2 – Permissive: [https://ohwr.org/project/cernohl/-/wikis/Documents/CERN-OHL-version-2]

[48] Licensee: [https://github.com/licensee/licensee]

[49] Shields.io: [https://shields.io]

[50] Hofmann, Frank, and Axel Beckert. *Debian-Paketmanagement-Buch* [*Debian Package Management Book*]: [https://dpmb.org/index.en.html] (in German)

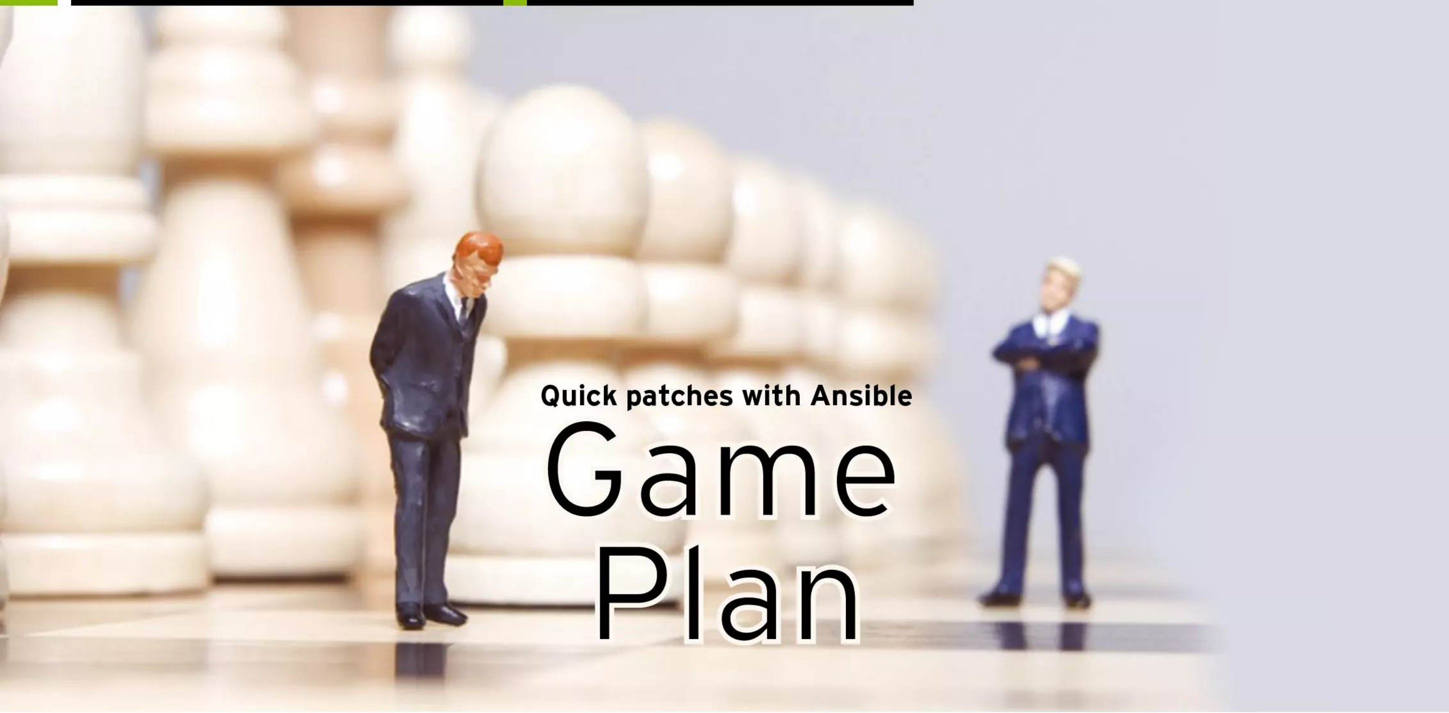
The Author

Frank Hofmann mostly works on the road as a developer, trainer, and author. His preferred workplaces are Berlin, Geneva, and Cape Town. He is one of the authors of the *Debian Package Management Book* [50].

Acknowledgements

The author thanks Veit Schiele, Axel Beckert, and Gerold Rupprecht for their suggestions during the preparation of this article.

■



Quick patches with Ansible

Game Plan

Ansible has earned an excellent reputation as a tool for configuration management. Thanks to playbooks, updates are more or less child's play. By Thomas Reuß

Nobody enjoys patching servers – or at least I don't know anyone who claims to – but everybody agrees that security patches are essential, at least for front-line systems such as web servers, firewalls, load balancers, mail servers, and so on. However, even in 2025, I still hear people saying (in essence): "You don't need to worry, you know. These machines are inaccessible from the Internet and hidden right at the back of the LAN."

Listing 1: YAML Inventory File

```
01 ---
02 servers:
03   children:
04     debian_servers:
05       hosts:
06         # Pi-hole
07         pihole:
08           ansible_ssh_private_key_file: /home/treuss/.ssh/pihole
09           ansible_user: thomas
10         # Gitea
11         raspi03:
12           ansible_ssh_private_key_file: /home/treuss/.ssh/raspi03
13           ansible_user: thomas
14         # Backup
15         raspi06:
16           ansible_ssh_private_key_file: /home/treuss/.ssh/raspi06
17           ansible_user: thomas
18     dns_servers:
19       hosts:
20         # Pi-hole
21         pihole:
22           ansible_ssh_private_key_file: /home/treuss/.ssh/pihole
23           ansible_user: thomas
```

This setup could even work for a while, until someone right at the back of the LAN has the bright idea of plugging the fancy USB stick they found earlier in the lounge into their work PC or of opening what appears to be a totally harmless email attachment – after all, the company has a painfully expensive state-of-the-art endpoint security solution in place. In these cases, even the most attractive, fancy software box on the shelf in the admin's office is of no use whatsoever. Do you have highly segmented networks in your back office? Are your LAN sockets protected by MAC filters? Do you disable USB ports or even lock away your hardware in metal boxes? The thought of users opening up a path for malware right into the heart of the company is likely to make many an admin break out in a cold sweat.

Against the backdrop of these horror scenarios, it will not hurt to look at some totally banal but still very real issues, including rolling out mass updates, which plays a particularly important role if you don't have access to powerful, but potentially expensive, tools like SUSE Multi-Linux Manager [1] or Uyuni [2]. In such a case, it's worth considering Ansible [3].

Inventory

Ansible always begins with inventory, irrespective of whether you are starting with an existing infrastructure or setting up a greenfield deployment. Ideally, the Ansible inventory will include all of your hosts at some point. I am deliberately not talking about all the critical hosts. If you take a closer look at this kind of automation, you will quickly understand that the less critical systems do not cause any additional overhead. A typical inventory file (Listing 1) can look relatively simple. Whether structured in YAML [4] or the INI format, it ultimately has no influence on the results. I opted for YAML. As you can see, I defined my hosts in the file (Figure 1). In this specific case, the inventory is named `servers.yml`. As the name suggests, it only contains servers. How you break down your hosts is up to you. In this example, I distinguished between `debian_servers` (line 4) and `dns_servers` (line 18). Although I have only been running Debian servers up to now, I would like to keep open the option of setting up a SUSE- or Gentoo-based machine. Another reason for classifying by (Layer 7) services is that I use

Lead Image © David Crockett, Fotolia.com

Pi-hole [5] as an ad blocker. It has its own update management, which has its own life outside the distribution package manager. If I want to update Pi-hole, I can't just call `apt update && apt upgrade`.

I defined two variables for each host: `ansible_user` specifies the user Ansible will use to log in to the target system, and `ansible_ssh_private_key_file` stipulates private key-based SSH login.

Playbooks

Besides inventory, the Ansible world has other important files. I'm talking about playbooks, of course. Although the inventory determines the subject of the operation, playbooks define what will happen.

The sample playbook in Listing 2 for updating Debian systems follows a relatively simple structure, much like the inventory example. Line 2 contains the playbook's name, and line 3 ensures that Ansible only executes the tasks that follow on my Debian servers.

Listing 2: Ansible Playbook

```
01 ---
02 - name: Patch Debian servers
03   hosts: debian_servers
04   tasks:
05   - name: Use apt to update all Debian servers
06     ansible.builtin.apt:
07       force_apt_get: yes
08       upgrade: dist
09     become: yes
10     become-method: sudo
```

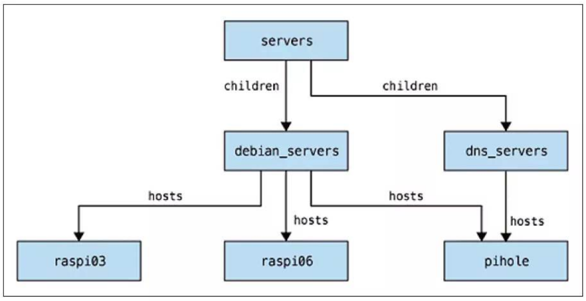


Figure 1: The schematic of an Ansible inventory showing the hosts and connections.

Experience has shown that the YAML format is rarely self-explanatory for newcomers. For deeper insights, please refer to the documentation [5]. For my purposes, a couple of brief details should suffice: Ansible interprets lines beginning with a dash as list elements. Lines of the form `<key>: <value>` are the key-value pairs of a dictionary or a hash table. In contrast to long-established formats such as XML or JSON, whitespace is very important in YAML. As in Python, the indentation depth determines to which element a line belongs. Complex structures can be created with just these few syntax elements.

This playbook, named *Patch Debian servers* in line 2 targets all the hosts in the inventory that belong to the `debian_servers` host group. Ansible applies the specified `tasks` to each of these hosts. This example has just one task, `ansible.builtin.apt`, and its assigned name in line 5 is *Use apt to update all Debian servers*. Line 6, `ansible.builtin`, already suggests something from the tool's standard repertoire: the built-in Apt module.

Linux users do not laboriously have to gather the tools needed for their choice of operating system. Ansible comes with a massive collection out of the box. The `force_apt_get` switch forces Apt to update the package sources again. The `upgrade` switch tells Apt to install the available updates. Finally, I used the `--become` and `--become-method` parameters to tell Ansible that a context switch to root is required, which relies on `sudo`.

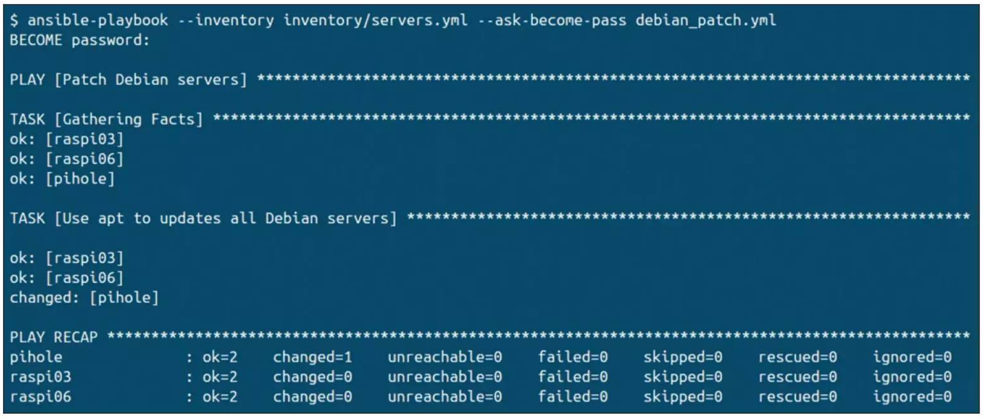
If everything is correct, and the indentations in the YAML files match, the output you can see in Figure 2 is gradually generated when the code

```
$ ansible-playbook ➔
  --inventory inventory/servers.yml ➔
  --ask-become-pass debian_patch.yml
```

is called. Ansible always determines the facts for all hosts first, which in this case means extensive information, including all the IP addresses, hostnames, and much more.

In the *Use apt to update all Debian servers* task, Ansible just says *ok* for two hosts, *raspi03* and *raspi06*. Therefore, no updates were available by Apt, so Ansible did not have anything to do. No changes occurred in the case of Pi-hole. In Ansible parlance, a change means a deviation from the target state, which the tool was able to restore.

The `--ask-become-pass` option tells software to request the password interactively for the context switch to root from the user. Ansible again



```
$ ansible-playbook --inventory inventory/servers.yml --ask-become-pass debian_patch.yml
BECOME password:

PLAY [Patch Debian servers] *******************************************************************************

TASK [Gathering Facts] ************************************************************************************
ok: [raspi03]
ok: [raspi06]
ok: [pihole]

TASK [Use apt to updates all Debian servers] **************************************************************

ok: [raspi03]
ok: [raspi06]
changed: [pihole]

PLAY RECAP ************************************************************************************************
pihole                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
raspi03               : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
raspi06               : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figure 2: If the YAML file has no errors – including indentation – Ansible successfully runs the playbook.

Listing 3: Updating Pi-hole

```
01 ---
02 - name: Update Pi-hole server
03   hosts: dns_servers
04   tasks:
05   - name: Update Pi-hole software
06     ansible.builtin.command: pihole updatePihole
07     become: yes
08   - name: Update Gravity (AdLists)
09     ansible.builtin.command: pihole UpdateGravity
10     become: yes
```

offers tools for this use case: vaults. With the help of vaults, you can also run playbooks without password prompts.

Ansible is a tool for configuration management, so ultimately you specify a target state in the inventory and some other files later on; the Ansible modules then try to establish this state within the framework of playbooks. In other words, if Apt does not find any updatable packages, the Debian systems must already be in the target state, which is a good thing. In the case of `raspi03` and `raspi06`, I activated `unattended-upgrades` for the hosts at setup. Both systems update automatically on a regular basis, which is why no updates needed to be installed when I called them in Figure 2.

Thinking Ahead

System updates are not all the work that needs to be done. If you run your own server, you will be aware that you cannot always simply download the software you need from your favorite distribution's package repository. Typical examples include ownCloud, Nextcloud, Gitea, Mastodon, and in this case, Pi-hole. Of course, I always want to have the latest version of the software with the latest ad lists containing URLs from advertisers, which means that it made sense to write my own playbook for updating Pi-hole [5].

Ultimately, just two commands are all it took, starting with

```
sudo pihole updatePihole
sudo pihole updateGravity
```

to update the software and the lists of advertisers to be blocked. Both commands require root privileges, as you can see from the call to `sudo`. Therefore, you again need the `become` parameter at this point.

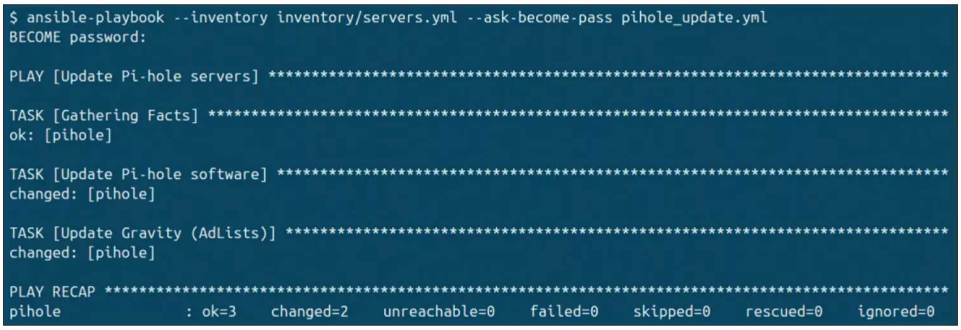
In Listing 3, you can see the two calls defined as two tasks. As a reminder, the hyphens stand for list elements, which is why any number of tasks can be strung together. This playbook tells Ansible to run the two commands on all `dns_servers` (line 3). I used the `ansible.builtin.command` (lines 6 and 9) to call shell commands without any serious configuration overhead. However, keep in mind that Ansible interprets a task as successful if the return code (RC) of a program or shell script is zero. Experienced shell users therefore do not need to change anything; the logic remains the same.

Figure 3 shows the output from the playbook. At first glance, you can see that Ansible always uses the same, tidy-looking output regardless of the tasks at hand. Ultimately, it outputs one line per host containing the number of successful tasks (*ok*), the number of changes (i.e., deviations from the target state), and so on.

Conclusions

Although the use of Ansible for simple tasks appears to be a little over the top at first glance, doing so offers massive benefits on closer inspection. You can update entire server landscapes in next to no time, and you can identify hosts on which an automatic update process in the form of the `unattended-upgrades` role could be useful. Automatically updating your own services – in my case Pi-hole – even ensures that you have documentation for the process. A correctly authored playbook clearly explains which steps to take – which is especially useful for applications that are not updated regularly.

Ansible can provide useful support for your everyday chores, but the tool really comes into its own when it takes on the massive task of installing new systems. The systems I am talking about here are servers with several SCSI controllers for a large number of logical volume data carriers, hosts for distributed databases, and application servers. Even if you have well-maintained templates, setting up these systems involves many hours of extremely error-prone work, always with the risk of forgetting some minute but critical detail. ■



```
$ ansible-playbook --inventory inventory/servers.yml --ask-become-pass pihole_update.yml
BECOME password:

PLAY [Update Pi-hole servers] *****************************************************************************

TASK [Gathering Facts] ************************************************************************************
ok: [pihole]

TASK [Update Pi-hole software] ****************************************************************************
changed: [pihole]

TASK [Update Gravity (AdLists)] ***************************************************************************
changed: [pihole]

PLAY RECAP ************************************************************************************************
pihole              : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figure 3: Ansible after updating Pi-hole. In the example, the tool successfully completed three tasks.

Info

[1] SUSE Multi-Linux Manager: [https://www.suse.com/products/multi-linux-manager/]

[2] Uyuni: [https://www.uyuni-project.org]

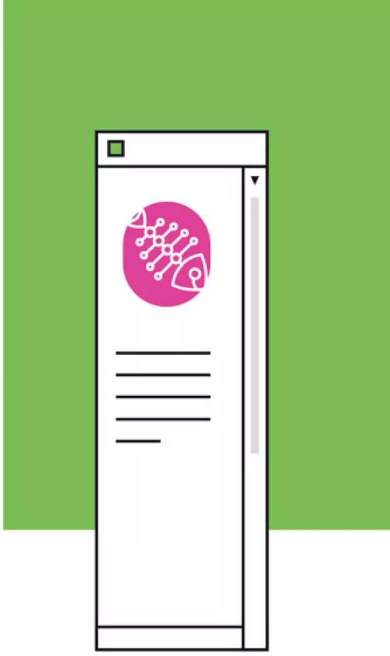
[3] Ansible documentation: [https://docs.ansible.com]

[4] YAML syntax: [https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html]

[5] Pi-hole documentation: [https://docs.pi-hole.net/core/pihole-command/]

Author

Thomas Reuß is a passionate Linux admin who is hugely interested in security. He is currently working as a consultant in the SAP environment.



open south code

2025

20-21 June

La Térmica
Málaga

technology and open culture

@opensouthcode

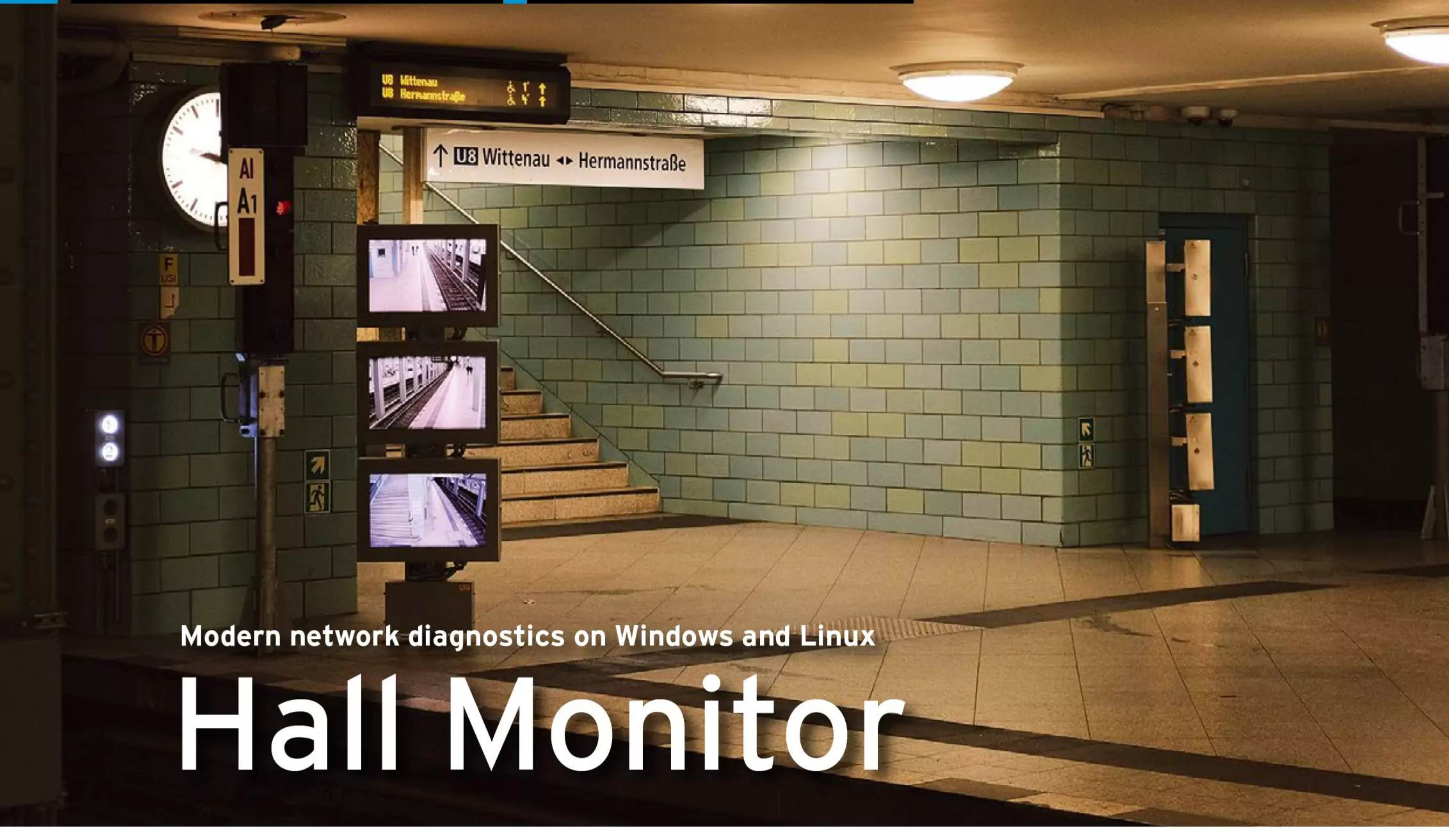
opensouthcode.org

Diamond
Canonical

Platinum
THE WORKSHOP
SECTOR ALARM TECH

Gold
FREEPIK
Ortus Solutions A Software Revolution
Deep Computing
spanred Telecomunicaciones
PICKY PRESSURE
VICTORIA MÁLAGA 1928
SLIMBOOK The best GNU/Linux computers

malaga.es
Diputación Provincial de Málaga
la térmica CENTRO DE CULTURA CONTEMPORÁNEA



Modern network diagnostics on Windows and Linux

Hall Monitor

`Get-NetTCPConnection` and `ss` are efficient and powerful tools for monitoring network connections and identifying potential security threats on Windows and Linux. By Andrea Ciarrocchi

In the evolving landscape of system administration and network diagnostics, legacy tools often give way to more efficient, modern alternatives. One such shift involves the deprecation of the widely used `netstat` command on both Windows and Linux systems. Although `netstat` served as a cornerstone for analyzing network connections, it has become outdated, with its functionality being replaced by more powerful and flexible tools: `Get-NetTCPConnection` on Windows and `ss` (socket statistics) on Linux. These new tools are not merely replacements but advancements designed to address the limitations of their predecessor, offering improved performance, greater precision, and enhanced integration with modern systems.

The Linux `ss` command provides a faster and more detailed way to view socket information than `netstat`. By directly accessing kernel data structures, `ss` delivers real-time insights into active connections, listening ports, and protocol-specific details. This efficiency makes it particularly useful for monitoring high-volume servers or troubleshooting complex networking scenarios. Moreover, `ss` offers a rich set of filtering options that allow you to hone in on specific connections, states, or protocols with minimal effort.

On Windows, the `Get-NetTCPConnection` cmdlet brings the power of PowerShell's object-oriented paradigm to network diagnostics. As part of the *NetTCPIP* module, it not only replicates the `netstat` core functionality but also integrates seamlessly into the broader PowerShell ecosystem, enabling advanced filtering, automation, and reporting. Unlike `netstat`, which outputs plain text, `Get-NetTCPConnection` provides structured data that can be easily manipulated and combined with other PowerShell cmdlets to create robust workflows. This modern approach makes it an indispensable tool for Windows administrators tasked with managing network resources and ensuring system security.

The Linux ss Command

The `ss` command is a powerful utility for examining network connections, sockets, and their statuses. It provides detailed insights into the state of your system's network, making it an essential tool for network diagnostics and troubleshooting.

To use `ss`, open a terminal and execute the command with the appropriate options. By default, running `ss` without options displays a summary of established network connections, including their source and destination addresses, ports, and states (Figure 1).

If you need more specific information, you can use the `-t`, `-u`, and `-x` options to filter by protocol. The `-t` flag restricts the output to TCP connections, whereas `-u` displays only UDP connections. The `-x` option shows Unix domain sockets, which are used for interprocess communication. For example, running `ss -t` lists only TCP connections, making it easier to focus on this protocol.

One of the most commonly used flags is `-n`, which prevents the resolution of IP addresses into hostnames and ports into service names, ensuring faster execution and numerical output. Conversely, the `-r` option resolves IPs into hostnames, which can help when understanding external connections but might slow down the

Photo by Nk Ni on Unsplash

command because of DNS lookups. For security audits, the `-p` flag is essential because it links sockets to the process names and IDs responsible for the connections, allowing users to trace suspicious activity back to specific applications. Advanced users might also find the `-i` option useful, because it reveals detailed internal socket information for debugging purposes.

To view listening sockets, you can use the `-l` option. For instance, `ss -lt` shows only listening TCP sockets, which is useful for checking which services or applications are awaiting incoming connections on specific ports. When you need to display process information alongside the sockets, adding the `-p` option reveals the process name and process ID (PID) associated with each connection. For example, `ss -tulpn` combines multiple options to display TCP and UDP listening sockets with associated processes and numeric port numbers.

Filtering capabilities in `ss` allow you to narrow down results on the basis of specific criteria. You can use keywords such as `dst` (destination) or `src` (source) followed by an IP address, port number, or hostname. For example,

```
ss dst 192.168.1.1
```

lists all connections targeting that IP address, whereas

```
ss src :80
```

displays connections originating from port 80.

For more detailed output, the `-i` option adds interface information, such as network interfaces involved in connections, and the `-e` option provides extended information about TCP sockets, like retransmission counts.

If you need continuous monitoring, adding `watch`,

```
watch -n 1 ss -t
```

refreshes the output at regular intervals, so you can observe real-time changes.

Filtering connections by state is another powerful feature of `ss`. For example, viewing established connections (`-e`) allows you to focus solely on active connections, or you can isolate sockets in the listening state (`-l`). These options can be further refined by the `-4` or `-6` options to limit the results to IPv4 or IPv6 connections. The `-f` flag lets users specify an address family (e.g., `inet` for IPv4 and IPv6 or `unix` for Unix domain sockets), providing a tailored view of network activity.

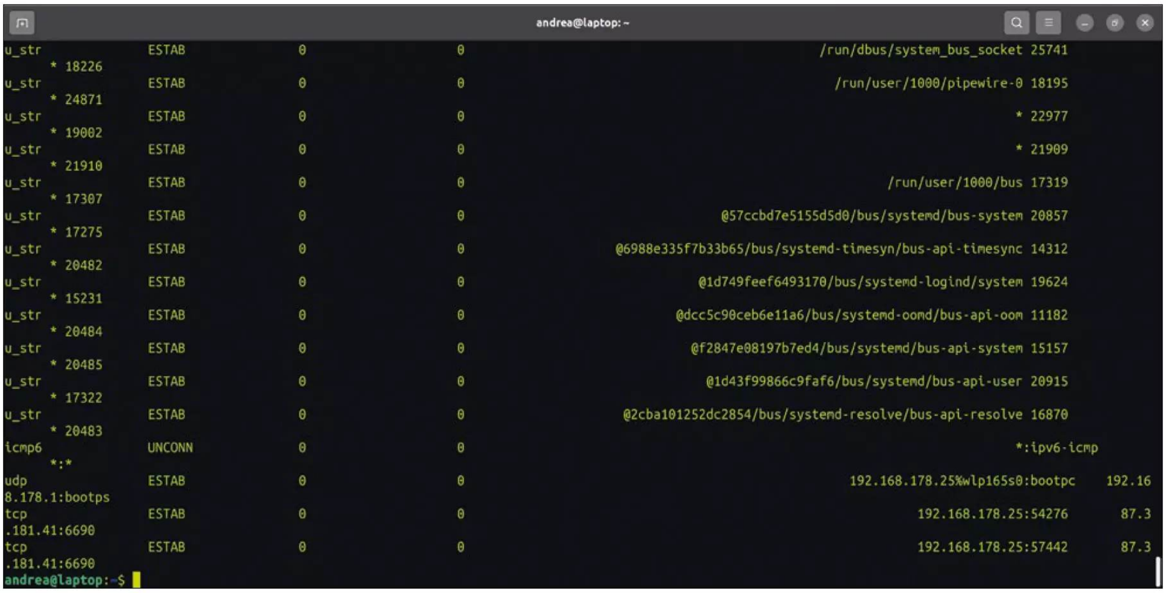
Identifying Potential Threats

When analyzing network connections with the `ss` command, the goal is to gain a deep understanding of the state of the system's network activity and identify any unusual patterns that could suggest malicious or unauthorized behavior. The `ss` command can produce a detailed view of the sockets on your system, helping you detect any irregularities.

To begin, you might run a simple

```
ss -tuln
```

command to display a list of all listening TCP and UDP sockets. The output includes the address and port combinations on which your system is awaiting incoming connections. Looking closely at this output, you should be familiar with what normal, expected services are running on your machine. For instance, you might see services like SSH on port 22, HTTP on port 80, or HTTPS on port 443. However, unexpected services listening on unusual ports could be a sign of a misconfiguration or even a potential security issue. For example, if a service like FTP is running without your knowledge or an unknown process is listening on a port that doesn't match any known service, you might want to investigate further.



```
u_str               ESTAB               0                    0                                                 /run/dbus/system_bus_socket 25741
       * 18226
u_str               ESTAB               0                    0                                                   /run/user/1000/pipewire-0 18195
       * 24871
u_str               ESTAB               0                    0                                                                           * 22977
       * 19002
u_str               ESTAB               0                    0                                                                           * 21909
       * 21910
u_str               ESTAB               0                    0                                                          /run/user/1000/bus 17319
       * 17307
u_str               ESTAB               0                    0                                    @57ccbd7e5155d5d0/bus/systemd/bus-system 20857
       * 17275
u_str               ESTAB               0                    0                      @6988e335f7b33b65/bus/systemd-timesyn/bus-api-timesync 14312
       * 20482
u_str               ESTAB               0                    0                                 @1d749feef6493170/bus/systemd-logind/system 19624
       * 15231
u_str               ESTAB               0                    0                              @dcc5c90ceb6e11a6/bus/systemd-oomd/bus-api-oom 11182
       * 20484
u_str               ESTAB               0                    0                                @f2847e08197b7ed4/bus/systemd/bus-api-system 15157
       * 20485
u_str               ESTAB               0                    0                                  @1d43f99866c9faf6/bus/systemd/bus-api-user 20915
       * 17322
u_str               ESTAB               0                    0                       @2cba101252dc2854/bus/systemd-resolve/bus-api-resolve 16870
       * 20483
icmp6               UNCONN              0                    0                                                                           *:ipv6-icmp
       *:*
udp                 ESTAB               0                    0                                                    192.168.178.25%wlp165s0:bootpc     192.16
8.178.1:bootps
tcp                 ESTAB               0                    0                                                              192.168.178.25:54276       87.3
.181.41:6690
tcp                 ESTAB               0                    0                                                              192.168.178.25:57442       87.3
.181.41:6690
andrea@laptop:~$ 
```

Figure 1: The `ss` command output on Linux.

Another key aspect of monitoring is checking the status of the active connections. The

```
ss -tn
```

command lets you examine the current state of active TCP connections (Figure 2). Each connection is shown with its status, which could be *ESTABLISHED*, *SYN_SENT*, *TIME_WAIT*, or other states that indicate the lifecycle of a connection. An unusually high number of connections in certain states can indicate abnormal behavior. For example, a surge in *SYN_SENT* connections might suggest a SYN flood attack, which is a type of denial-of-service attack that attempts to overwhelm the target system by sending many connection requests that go unanswered. If you're seeing many connections stuck in this state, it could point to this kind of attack. Similarly, if an unexpected number of connections are in the *TIME_WAIT* state, you might have a problem with connection teardown, potentially because of a malfunctioning service or attack.

Beyond just the connection state, it's critical to examine which processes are tied to each connection. The

```
ss -p
```

command displays the PID and the name of the program associated with each connection. This information is useful for identifying suspicious activity tied to processes you might not recognize. For example, if you notice that a connection on an unusual port is being held by a process you don't recognize, it's worth digging deeper to determine whether the process is legitimate or whether it could be malware. On a secure system, you should confidently be able to identify the processes that have network connections. Anything outside of this norm is a red flag and should be investigated further.

An additional method for identifying unusual connections is to monitor the system's network activity over time. Regularly running `ss` with the appropriate options allows you to spot trends and changes in network behavior. For example, if you regularly run `ss` and suddenly notice a significant spike in the number of connections or an influx of foreign IP addresses connecting to your system, it could indicate an unauthorized access attempt or data exfiltration. Similarly, connections to unusual ports, such as those not typically used by your services, should be flagged for investigation.

Cross-referencing the IP addresses you see in the connection list is also important. If you spot IP addresses from regions or countries that you don't expect to communicate with your system, or if the IP addresses are known to be associated with malicious activity (on the basis of threat intelligence feeds or a reputable IP blacklist), you have a strong indicator that something might be wrong.

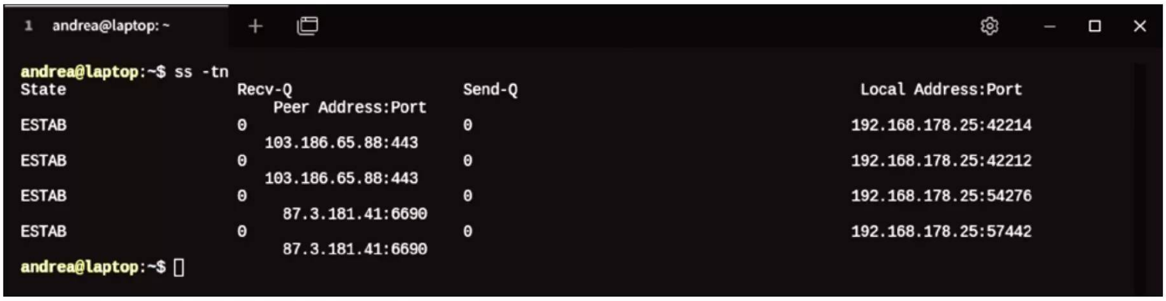
The beauty of the `ss` command lies in the granularity of the information it provides. By observing patterns, process associations, and the state of network connections, you can develop a thorough understanding of your system's network activity, which over time, enables the recognition of what constitutes normal behavior, making it easier to spot irregularities that could signify potential security threats. By leveraging `ss` as part of a broader network monitoring strategy, you can proactively detect issues and take steps to safeguard your system.

Get-NetTCPConnection Windows Cmdlet

The `Get-NetTCPConnection` command in Windows PowerShell is a modern and powerful tool designed to retrieve detailed information about TCP connections on a Windows system. It is part of the *NetTCPIP* module. Unlike `netstat`, which outputs plain text that often requires additional parsing, `Get-NetTCPConnection` produces structured data that integrates seamlessly into PowerShell's object-oriented framework, making it far more flexible and suitable for modern network diagnostics and automation.

When executed without parameters, `Get-NetTCPConnection` displays all active TCP connections and listening ports on the system (Figure 3). The default output includes essential details such as the local and remote IP addresses, their corresponding ports, the state of the connection, and the PID of the application responsible for each connection. For example, you might see connections in an established state, which indicates active communication, or a listening state, which shows that a port is awaiting incoming connections.

The command can also filter and refine the output by specific criteria.



```
andrea@laptop:~$ ss -tn
State                Recv-Q               Send-Q                     Local Address:Port
                       Peer Address:Port
ESTAB                0                    0                          192.168.178.25:42214
                     103.186.65.88:443
ESTAB                0                    0                          192.168.178.25:42212
                     103.186.65.88:443
ESTAB                0                    0                          192.168.178.25:54276
                       87.3.181.41:6690
ESTAB                0                    0                          192.168.178.25:57442
                       87.3.181.41:6690
andrea@laptop:~$ 
```

Figure 2: The current state of active Linux TCP connections.

For instance, if you want to view only established connections, you can use the `-State` parameter and specify `Established`. Similarly, you can narrow the results to connections associated with a specific local port, such as port 80, with the `-LocalPort` parameter. If you need to investigate connections to a particular remote IP address or port, the `-RemoteAddress` and `-RemotePort` parameters let you isolate these details with precision.

By examining the `OwningProcess` property, which provides the PID of the application managing a connection, you can identify the responsible program. This information is particularly useful when troubleshooting connectivity issues or detecting unexpected network activity. For a more detailed analysis, the results can be enriched by combining `Get-NetTCPConnection` with other PowerShell cmdlets like `Get-Process`. For example, you can append the process name associated with each connection to the output, offering a clearer view of how applications are interacting with the network.

The structured nature of the `Get-NetTCPConnection` output also allows for easy export and further analysis. By piping the command's results into tools like `Export-Csv`, you can generate reports that are suitable for audits or compliance purposes. Additionally, because PowerShell allows for real-time filtering and scripting, you can create automated workflows that continuously monitor your system's connections, alerting you to suspicious activity, or generate periodic summaries.

Identifying Potential Threats

`Get-NetTCPConnection` can be used to identify unusual or suspicious connections in much the same way you would use `ss` on Linux.

To obtain similar information on Windows, you can use `Get-NetTCPConnection` to display details about the active TCP connections and their associated states. By default, `Get-NetTCPConnection` shows connections in a variety of states (e.g., *Established*, *Listen*, *SynSent*, and *CloseWait*). The `State` field is crucial for spotting unusual patterns that might indicate an attack or abnormal behavior, like an excessive number of connections in a *SYN_SENT* state, which could suggest a SYN flood attack.

If you want to see all active TCP connections, you would run the PowerShell command

```
Get-NetTCPConnection
```

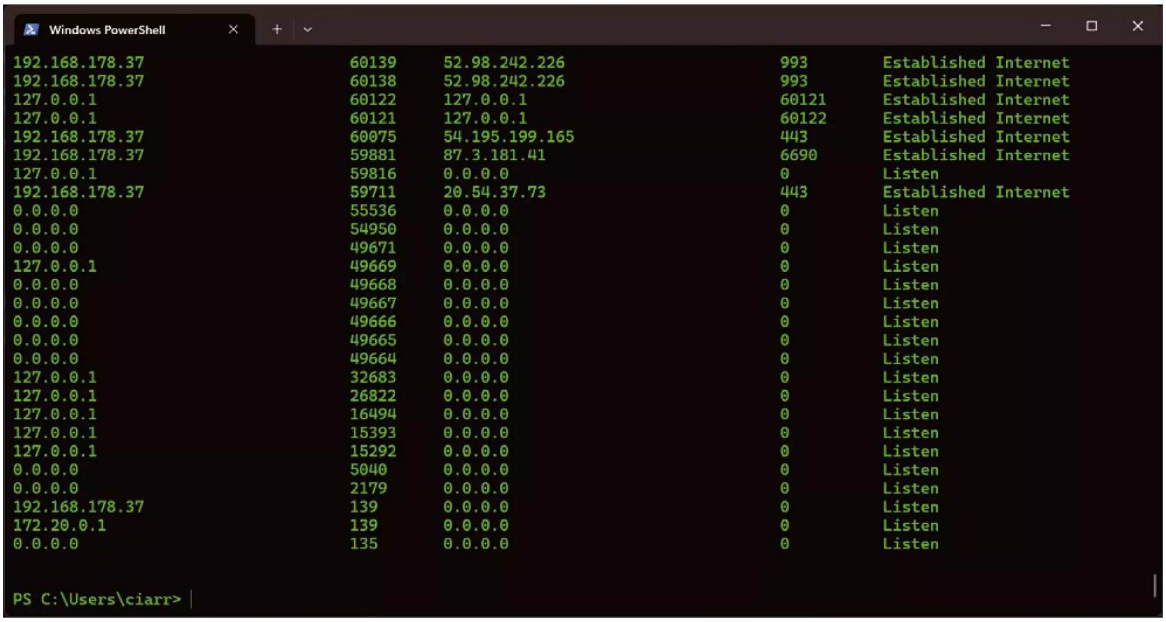
which lists all TCP connections, including information such as the local address, local port, remote address, remote port, and state of the connection. To filter the results by a specific state or look for connections in the *Established* state, you could use the command

```
Get-NetTCPConnection | ↩
  Where-Object { $_.State -eq ↩
                'Established' }
```

which returns only the currently established connections, helping you identify active connections to and from your system.

If you're interested in seeing which process is associated with each connection, the `Get-NetTCPConnection` cmdlet reveals the owning PID. By correlating the PID with running processes (with the `Get-Process` cmdlet), you can determine which program is using a specific connection. To display this information, run,

```
Get-NetTCPConnection | ↩
 Select-Object LocalAddress, LocalPort, ↩
              RemoteAddress, RemotePort, ↩
              State, OwningProcess
```



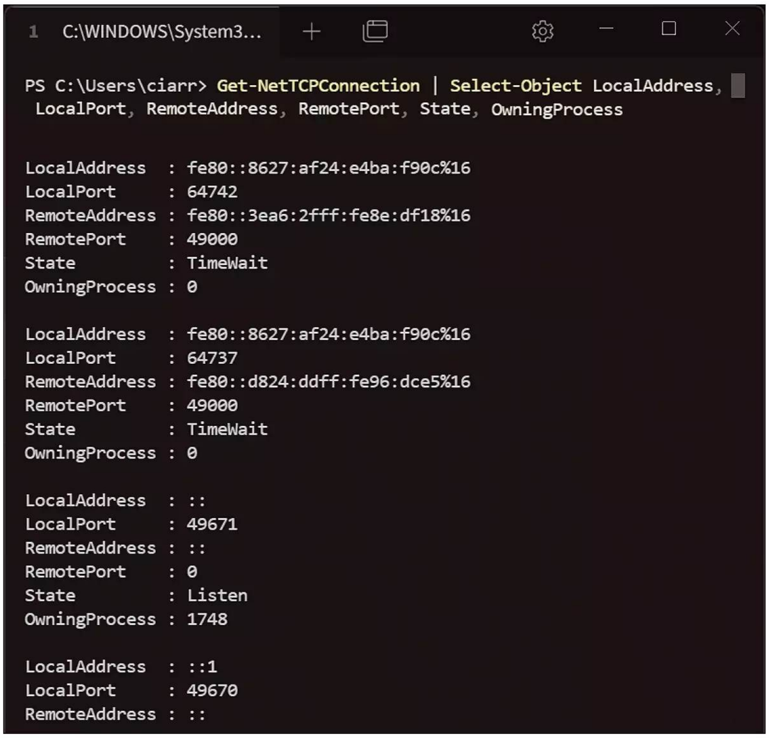
```
192.168.178.37                    60139    52.98.242.226                    993       Established Internet
192.168.178.37                    60138    52.98.242.226                    993       Established Internet
127.0.0.1                         60122    127.0.0.1                        60121     Established Internet
127.0.0.1                         60121    127.0.0.1                        60122     Established Internet
192.168.178.37                    60075    54.195.199.165                   443       Established Internet
192.168.178.37                    59881    87.3.181.41                      6690      Established Internet
127.0.0.1                         59816    0.0.0.0                          0         Listen
192.168.178.37                    59711    20.54.37.73                      443       Established Internet
0.0.0.0                           55536    0.0.0.0                          0         Listen
0.0.0.0                           54950    0.0.0.0                          0         Listen
0.0.0.0                           49671    0.0.0.0                          0         Listen
127.0.0.1                         49669    0.0.0.0                          0         Listen
0.0.0.0                           49668    0.0.0.0                          0         Listen
0.0.0.0                           49667    0.0.0.0                          0         Listen
0.0.0.0                           49666    0.0.0.0                          0         Listen
0.0.0.0                           49665    0.0.0.0                          0         Listen
0.0.0.0                           49664    0.0.0.0                          0         Listen
127.0.0.1                         32683    0.0.0.0                          0         Listen
127.0.0.1                         26822    0.0.0.0                          0         Listen
127.0.0.1                         16494    0.0.0.0                          0         Listen
127.0.0.1                         15393    0.0.0.0                          0         Listen
127.0.0.1                         15292    0.0.0.0                          0         Listen
0.0.0.0                           5040     0.0.0.0                          0         Listen
0.0.0.0                           2179     0.0.0.0                          0         Listen
192.168.178.37                    139      0.0.0.0                          0         Listen
172.20.0.1                        139      0.0.0.0                          0         Listen
0.0.0.0                           135      0.0.0.0                          0         Listen

PS C:\Users\ciarr>
```

Figure 3: `Get-NetTCPConnection` output on Windows 11.

which shows the local and remote addresses, the ports involved, the state of the connection, and the PID for each connection (Figure 4). Once you have the PID, you can further investigate the associated process:

```
Get-Process -Id <PID>
```



```
PS C:\Users\ciarr> Get-NetTCPConnection | Select-Object LocalAddress,
 LocalPort, RemoteAddress, RemotePort, State, OwningProcess

LocalAddress  : fe80::8627:af24:e4ba:f90c%16
LocalPort     : 64742
RemoteAddress : fe80::3ea6:2fff:fe8e:df18%16
RemotePort    : 49000
State         : TimeWait
OwningProcess : 0

LocalAddress  : fe80::8627:af24:e4ba:f90c%16
LocalPort     : 64737
RemoteAddress : fe80::d824:ddff:fe96:dce5%16
RemotePort    : 49000
State         : TimeWait
OwningProcess : 0

LocalAddress  : ::
LocalPort     : 49671
RemoteAddress : ::
RemotePort    : 0
State         : Listen
OwningProcess : 1748

LocalAddress  : ::1
LocalPort     : 49670
RemoteAddress : ::
```

Figure 4: Viewing details for each connection with `Get-NetTCPConnection` on Windows 11.

This information lets you track down any suspicious or unknown processes that might be associated with unusual network connections, much as you would use `ss -p` on Linux.

To filter out specific connections by remote address, local port, or other criteria, you can adjust the `Where-Object` filter. For example, if you wanted to see all connections on port 80 (HTTP), you could run

```
Get-NetTCPConnection | ↩
  Where-Object { $_.LocalPort -eq 80 }
```

to filter the connections to show only those on port 80. This information might be useful if you're monitoring web traffic.

Conclusion

Both `ss` on Linux and `Get-NetTCPConnection` on Windows offer powerful capabilities for monitoring network connections and identifying potential security threats. By leveraging these tools, system administrators and security professionals can gain valuable insights into the state of active connections, detect abnormal patterns, and quickly identify processes that could be linked to malicious activity. ■

Author
Andrea Ciarrocchi is a technology enthusiast. Visit his homepage at https://andreaciarrocchi.altervista.org.

CLEAR OFF YOUR BOOKSHELF WITH DIGITAL ARCHIVES

Complete your collection of *Linux Magazine* and *ADMIN Network & Security* with our Digital Archive Bundles.

You get a full year of issues in PDF format to access at any time from any device.

2024 Archives Available Now!

shop.linuxnewmedia.com/shop

Image © rukanoga, 123RF.com

Managing FOSS applications on AIX

Tools at the Ready!

Starting your journey with open source software on AIX has never been easier with the IBM AIX Toolbox. By Chris Gibson

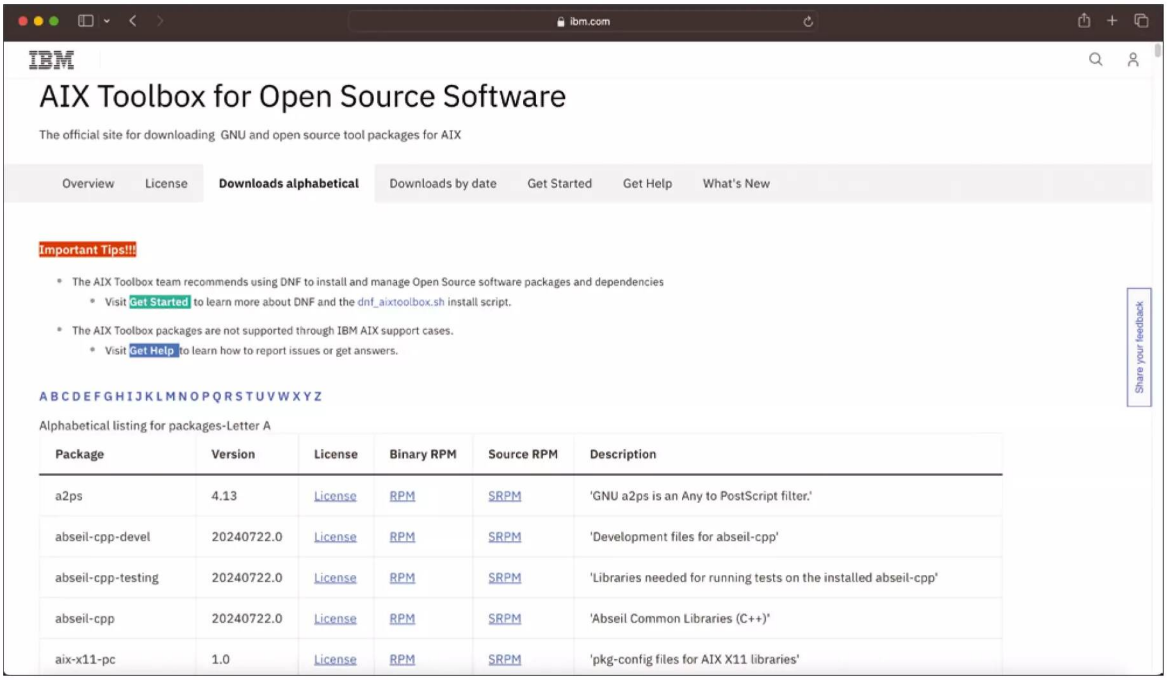
IBM has taken an active role in free and open source software (FOSS), in particular for Linux running on their Power servers. All modern day Power servers are now capable of running many different Linux distributions natively. Great, but what about the classic AIX operating system that still runs on so much IBM hardware?

Over the last couple of decades, IBM has taken steps to build open support into AIX [1]. The AIX Toolbox for Open Source Software [2] is an IBM site that serves as the primary source of GNU and open source tool packages ported to the AIX environment. The focus of this article is to look at the most efficient method of managing FOSS on AIX with the DNF package manager.

AIX Toolbox

Today more than 400 open source packages are available for download at the IBM hosted repository (Figure 1). The packages are kept current, with an update provided annually for most packages.

Any security vulnerabilities are addressed, and fixes are provided in a timely fashion. The AIX Toolbox contains many popular and useful development packages, such as languages, compilers, and debuggers. Many



AIX Toolbox for Open Source Software

The official site for downloading GNU and open source tool packages for AIX

Overview | License | Downloads alphabetical | Downloads by date | Get Started | Get Help | What's New

Important Tips!!!

- The AIX Toolbox team recommends using DNF to install and manage Open Source software packages and dependencies
  - Visit Get Started to learn more about DNF and the dnf_aixtoolbox.sh install script.
- The AIX Toolbox packages are not supported through IBM AIX support cases.
  - Visit Get Help to learn how to report issues or get answers.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Alphabetical listing for packages-Letter A

Package	Version	License	Binary RPM	Source RPM	Description
a2ps	4.13	License	RPM	SRPM	'GNU a2ps is an Any to PostScript filter.'
abseil-cpp-devel	20240722.0	License	RPM	SRPM	'Development files for abseil-cpp'
abseil-cpp-testing	20240722.0	License	RPM	SRPM	'Libraries needed for running tests on the installed abseil-cpp'
abseil-cpp	20240722.0	License	RPM	SRPM	'Abseil Common Libraries (C++)'
aix-x11-pc	1.0	License	RPM	SRPM	'pkg-config files for AIX X11 libraries'

Figure 1: The IBM AIX Toolbox website for open source software.

Lead Image © Nataliia Natykach, 123RF.com

foundational packages are also available, such as MariaDB, PostgreSQL, Apache, and more.

The AIX Toolbox contains a wide variety of third-party open source tools that can be installed on AIX by the RPM package manager (refer to the AIX Toolbox website [2] for more information). However, newer, more advanced tools are available to assist administrators with managing open source software (OSS) on AIX.

FOSS Management on AIX

By default, the `rpm` command is available on AIX (with the `rpm.rte` AIX fileset) to install, query, verify, and manage software packages, making it an essential tool for system administrators working with FOSS. However, `rpm` alone does not support automatic installation of prerequisites and does not automatically expand filesystems. DNF, on the other hand, helps with resolving RPM dependencies and prerequisites, but it's not installed with AIX by default, so I'll start with a close look at how you can install DNF on AIX. Then, I'll discuss how you can create and manage local DNF repositories, and I'll conclude by showing you an example of how to install a FOSS application on AIX with DNF.

DNF on AIX

DNF is an open source command-line package management utility for installing, removing, querying, and managing RPM packages. DNF automatically determines dependencies for the packages being updated or installed and will fetch the dependent packages and install them with the requested packages. The tool helps you avoid the well-known and feared RPM "dependency hell" problem [3]. DNF works with an existing software repository that contains RPM packages. The repository can be accessed locally or over a network. It's available to install on AIX versions 7.1, 7.2, and 7.3. The AIX Toolbox provides the required DNF install packages. I should mention that, yes, YUM was (is) also available on AIX; however, it's not supported on AIX 7.3 or later, so my advice for AIX is to stick with DNF, *not* YUM.

Before attempting to install DNF on AIX, you should first check that your system meets the minimum requirements. DNF operates on AIX 7.1 TL5 and later versions. The `openssl` fileset, version 1.1.X.X (or higher), must be installed. Ideally you'll have the latest version on your system. You must meet some disk space requirements for installing DNF, as well: at least 512MB of free space available in `/tmp`, `/opt`, and `/var`. If necessary, increase the amount of free disk space in each filesystem as needed before attempting installation.

If you miss some of the prerequisites, the DNF install script checks the system during installation and displays a message if any prerequisites are not met. Assuming you are ready to go, DNF can be installed directly from the Internet with a script available from the AIX Toolbox.

Installing DNF on AIX

If your AIX host has direct Internet access, the installation and setup of DNF is a breeze: You can use the `dnf_aixtoolbox.sh` script (from the AIX Toolbox PPC repository [4]). With this script, DNF is automatically configured to use the existing IBM AIX Toolbox repositories; that is, as part of the installation process, DNF is configured to access these repositories. Once DNF is installed, you do not need to make any changes to the configuration to use DNF with the IBM hosted repositories. The script downloads and installs the required `rpm.rte` version (if not already installed) and all the packages needed for DNF as a TAR bundle file.

The script has a couple of options, but my recommendation in most cases is to use the `-d:` flag, which installs and configures `dnf` if `yum` is not installed. Refer to the blog on the IBM Power community site for details on the available script flags [5].

As soon as DNF is installed, you should update root's `.profile` (or the `.kshrc` file, or both) to update the `$PATH` environment variable to include the location of the DNF (and other OSS) binaries in `/opt/freeware/bin` and `/opt/freeware/sbin`. The default DNF configuration file will point to the IBM hosted repositories. The repositories for AIX 7.3 are automatically configured in the `dnf.conf` file (Listing 1).

Once the root profile has been updated, `su` to root's updated environment and check that you can now list the available DNF repositories from

Listing 1: Default `dnf.conf`

```
[main]
cachedir=/var/cache/dnf
keepcache=1
debuglevel=2
logfile=/var/log/dnf.log
obsoletes=1
plugins=1
gpgcheck=1
installonly_limit=3
clean_requirements_on_remove=True
best=True
skip_if_unavailable=True

[AIX_Toolbox]
name=AIX generic repository
baseurl=https://anonymous:anonymous@public.dhe.ibm.com/
   aix/freeSoftware/aixtoolbox/RPMS/ppc/
enabled=1
gpgcheck=0

[AIX_Toolbox_noarch]
name=AIX noarch repository
baseurl=https://anonymous:anonymous@public.dhe.ibm.com/
   aix/freeSoftware/aixtoolbox/RPMS/noarch/
enabled=1
gpgcheck=0

[AIX_Toolbox_73]
name=AIX 7.3 specific repository
baseurl=https://anonymous:anonymous@public.dhe.ibm.com/
   aix/freeSoftware/aixtoolbox/RPMS/ppc-7.3/
enabled=1
gpgcheck=0
```

Listing 2: Listing DNF Repos

```
# echo "export PATH=$PATH:/opt/freeware/bin:/opt/
    freeware/sbin" >> /.profile
# su -
# dnf repolist
repo id                             repo name
AIX_Toolbox                         AIX generic
repository
AIX_Toolbox_73                      AIX 7.3 specific
repository
AIX_Toolbox_noarch                  AIX noarch repository
```

the IBM AIX Toolbox (Listing 2).

To ensure all the packages are up to date, run:

```
# dnf update
...etc...
Complete!
```

So far, so good as you can see from the output, but many internal AIX systems do not have Internet access for security reasons, in which case you have to download the DNF TAR bundle (`dnf_bundle_aix_73.tar`) manually from the AIX Toolbox PPC repository [6].

If you install DNF from the TAR bundle, the DNF configuration file will still point to the repositories located and accessible on the IBM public Internet server `public.dhe.ibm.com`. Figure 2 shows that several AIX 7.x systems have DNF installed by the install script; as a result, they access their DNF repositories over the Internet.

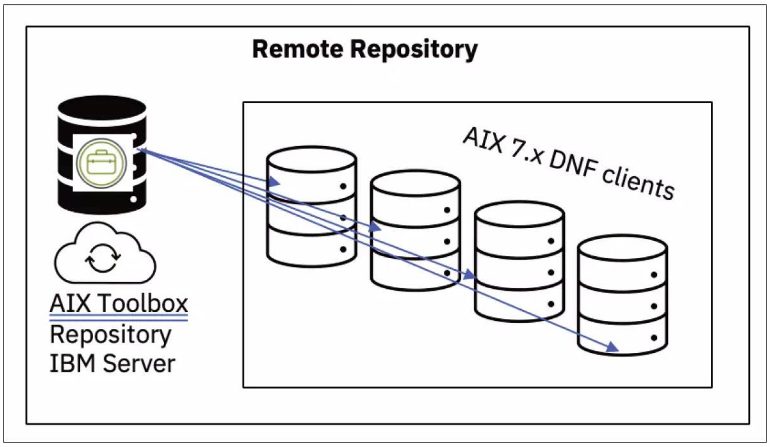


Figure 2: IBM hosted (remote) Internet DNF repository.

For an AIX host that doesn't have access to the Internet, this process is not very useful. A more practical solution is to provide access to the AIX Toolbox repositories from a local server.

Local Repositories

The AIX Toolbox site's directory structure can change; Figure 3 shows how it was organized at press time. The Toolbox can also be accessed from a web browser or with command-line tools like `lynx` or `wget` over HTTPS. However, the most common method of accessing the Toolbox is by DNF with local repositories.

Establishing local DNF repositories containing AIX Toolbox RPM packages is common practice in most organizations because of security policies that prevent direct Internet access to IBM repositories. For AIX hosts that don't have Internet access, you can configure a local AIX Toolbox repository server, then mirror the remote (Internet) repository to a local server, and configure AIX DNF clients to use the local repository server, instead of the Internet-hosted repositories.

To set up a local DNF repository, Internet access is required on one system, which will serve as the repository for other AIX systems. In this scenario, the system with Internet access (creating the local DNF repository) is referred to as the repo server. The other AIX systems that use this repository are known as DNF clients. The repo server must be an RPM-based system (AIX or an RPM-based Linux). A web server should be operational on the repo server, with the repository directory accessible to DNF clients over HTTP or HTTPS, which facilitates DNF connections and retrieval of RPM packages. Figure 4 shows a local repo server that has mirrored the Toolbox locally and DNF clients that access the local repo server over HTTP or HTTPS.

In this scenario, the `/opt/freeware/etc/dnf/dnf.conf` file on the Internet-connected repo server must have entries for the public IBM repository servers; then, the repo server must

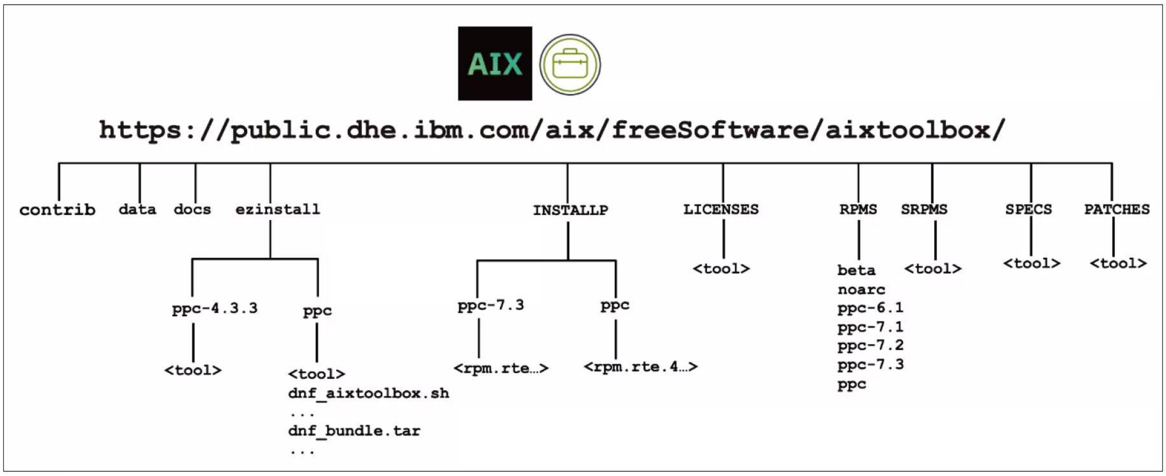


Figure 3: AIX Toolbox repository site structure.

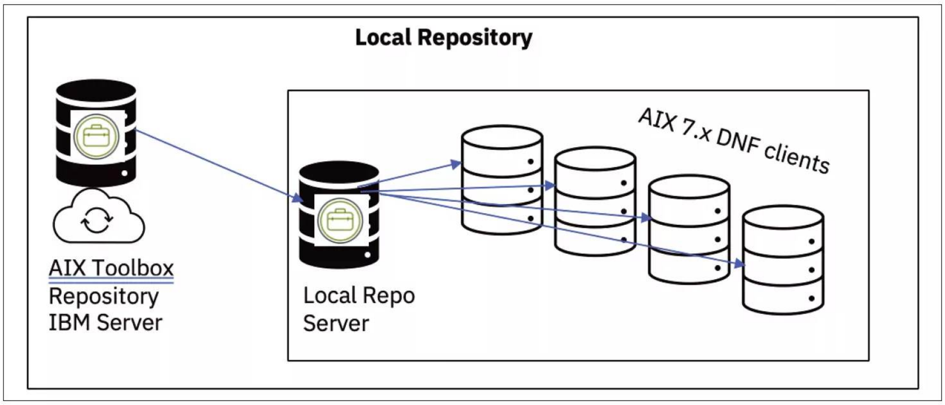


Figure 4: Locally hosted DNF repository.

sync (mirror) the entire AIX Toolbox repository with the `reposync` command, which mirrors the entire repository from the IBM servers to the local repo server.

The next step is to create the repository metadata because the `reposync` command simply mirrors all the files to the local server; it does not create the required repo metadata files. To do this on the newly synced local repos, you must run the `createrepo` command for each repository location that was synced. This method is not the focus of this article, so please refer to an article on the *IBM Developer* site [7] that provides a detailed description of the steps.

Now that the local repo server is configured and the repos are available, the DNF clients can be configured by (first installing DNF, if it is not already installed and then) editing their `dnf.conf` files and adding entries for the local repo server. A snippet from a sample `dnf.conf` file is shown in Listing 3, where the local repo server named `sisko` has a web server listening on port 80 (HTTP).

However, what if your local repo server also doesn't have Internet access? What can you do in this situation?

Creating a Local Repository

Setting up a local repository for a system with no Internet access (of any kind, from any AIX server) is possible and typically allowed by an organization's security policy. Two options are available in this case; however, in my opinion, one approach is more viable than the other.

In the first option, all of the RPM packages can be downloaded manually from the AIX Toolbox website; then, the administrator could copy the packages to a local repo server and configure the repositories. This time-consuming and tedious task would need to be performed each time the AIX Toolbox remote repository was updated, which requires at least one Internet-connected system somewhere in the organization that is able to download the *entire* AIX Toolbox. Often administrators will use their local PC or laptop to achieve this task (or even use the `wget -m` command from their Linux or Mac laptop). They then have to copy the files from their PC or laptop to the local AIX or Linux repo server (typically with `scp` or `sftp` for file transfer).

The second option, which I discuss in more detail here, is to download the Toolbox media ISO image (or TAR file) from IBM and then install the packages by either (1) mounting the ISO image or (2) extracting the packages from the TAR file. Of course, this method requires at least one Internet-connected system somewhere in the organization that is able to download the ISO image or TAR file from the IBM site and transfer the file to a local repo server. Again, sys admin's will typically use their local PC or laptop to achieve this task; then, they can copy the files from their workstation to the local AIX or Linux repo server (again, with `scp` or `sftp` for file transfer).

With available AIX Toolbox media images, you can set up a local DNF repository for AIX versions 7.1, 7.2, or 7.3. The Toolbox media provides a script to set up the local DNF repositories, install and configure DNF (if required), and add local repositories.

The ISO (or TAR) images contain a complete (point in time) copy of the entire AIX Toolbox. The ISO image (alone) is typically anywhere from 8 to 10GB in size. The steps for creating local repositories from media are as follows:

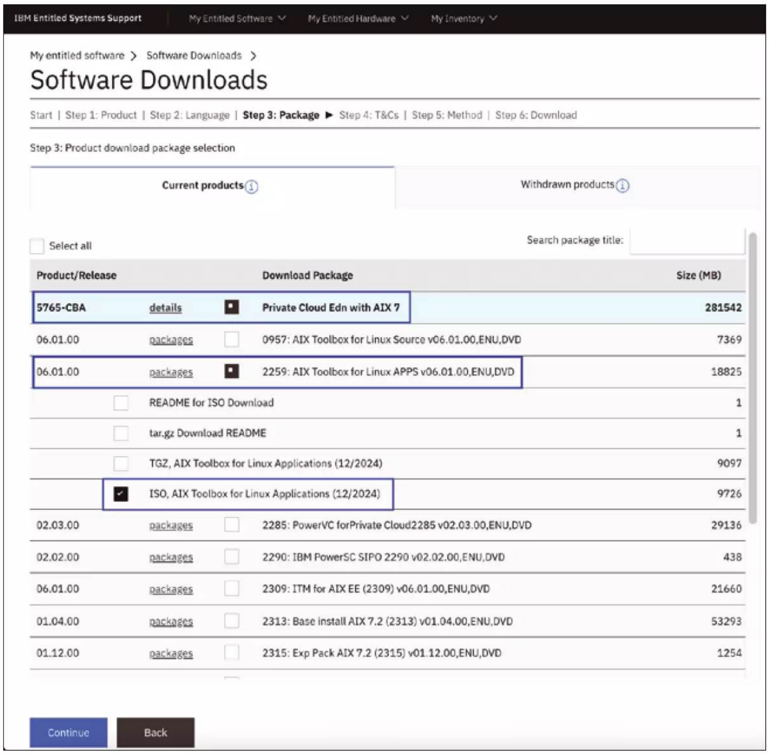
First, download the AIX Toolbox media ISO (or TAR or TGZ file) from either the IBM Entitled Systems Support (ESS) site [8] or the IBM Passport Advantage website [9] (depending on which AIX product authorizations you have).

If your source is ESS, you need to select and download either the ISO or the `tar.gz` (TGZ) image from the 2259 feature of your AIX product entitlement. If you're downloading from the Passport Advantage website, you need to select and download the file associated with one of the current parts M04S8ML (`.tar.gz`) or M04SBML (`.iso`). From here, look for the most recent AIX eAssembly and download the file for one of two entries with labels that include *AIX Toolbox for Linux Applications*.

I assume you are downloading the ISO image from ESS (Figure 5). Under *Private Cloud Edn with AIX*

Listing 3: `dnf.conf` for Local Repo Server

```
[My_AIX_Toolbox]
name=My AIX generic repository
baseurl=http://sisko:80//aixtoolbox/RPMS/ppc
enabled=1
gpgcheck=0
...etc...
```



IBM Entitled Systems Support | My Entitled Software | My Entitled Hardware | My Inventory

My entitled software > Software Downloads >

Software Downloads

Start | Step 1: Product | Step 2: Language | **Step 3: Package** ► Step 4: T&Cs | Step 5: Method | Step 6: Download

Step 3: Product download package selection

Current products | Withdrawn products

☐ Select all

Search package title:

Product/Release			Download Package	Size (MB)
5765-CBA	details	☑	**Private Cloud Edn with AIX 7**	**281542**
06.01.00	packages	☐	0957: AIX Toolbox for Linux Source v06.01.00,ENU,DVD	7369
06.01.00	packages	☑	2259: AIX Toolbox for Linux APPS v06.01.00,ENU,DVD	18825
	☐		README for ISO Download	1
	☐		tar.gz Download README	1
	☐		TGZ, AIX Toolbox for Linux Applications (12/2024)	9097
	☑		ISO, AIX Toolbox for Linux Applications (12/2024)	9726
02.03.00	packages	☐	2285: PowerVC forPrivate Cloud2285 v02.03.00,ENU,DVD	29136
02.02.00	packages	☐	2290: IBM PowerSC SIPO 2290 v02.02.00,ENU,DVD	438
06.01.00	packages	☐	2309: ITM for AIX EE (2309) v06.01.00,ENU,DVD	21660
01.04.00	packages	☐	2313: Base install AIX 7.2 (2313) v01.04.00,ENU,DVD	53293
01.12.00	packages	☐	2315: Exp Pack AIX 7.2 (2315) v01.12.00,ENU,DVD	1254

Figure 5: **Download *ISO, AIX Toolbox for Linux Applications (12/2024)* from IBM ESS.**

7, select the *2259: AIX Toolbox for Linux APPS* option and the *ISO, AIX Toolbox for Linux Applications (12/2024)* package to download an ISO image named (at the time of writing) `ESD-Toolbox_for_Linux_Apps_Common_7.2-7.3_122024_LCD4107740.iso`; transfer this file from your PC to your AIX server, where you plan to host the local repository.

Listing 4: `dnf_aixtoolbox_local.sh`

```
# cd /aixtoolbox/ezinstall/ppc
# ./dnf_aixtoolbox_local.sh /aixtoolbox

Copying dnf_bundle_aix_73.tar to /tmp/dnf-
   20241003191846 .....

Extracting dnf_bundle_aix_73.tar ...
x ca-certificates-2023.2.60-0.aix7.1.ppc.rpm, 991206
   bytes, 1936 tape blocks
...etc...
  31:python3-gpg-1.13.1-64_3          #################
      ################ [100%]

dnf installed successfully.
Please run 'dnf update' to update packages to the
   latest level.

...etc...

Creating the dnf.conf file with required locale
repositories.
The default /opt/freeware/etc/dnf/dnf.conf has been
saved as /opt/freeware/etc/dnf/dnf.conf_local_bak
#
```

Once the ISO has been downloaded and transferred, the next step is to use the `loopmount` command to mount the ISO over a new directory (mountpoint) named `/aixtoolbox`:

```
# mkdir /aixtoolbox
# loopmount ↵
  -i ESD-Toolbox_for_Linux_Apps_Common_↵
     7.2-7.3_122024_LCD4107740.iso ↵
  -o "-V udfs -o ro" ↵
  -m /aixtoolbox
```

Listing 5: Config File for Local Repositories

```
#
...
[Local_AIX_Toolbox]
name=Local AIX generic repository
baseurl=file:///aixtoolbox/RPMS/ppc/
enabled=1
gpgcheck=0

[Local_AIX_Toolbox_noarch]
name=Local AIX noarch repository
baseurl=file:///aixtoolbox/RPMS/noarch/
enabled=1
gpgcheck=0

[Local_AIX_Toolbox_73]
name=Local AIX 7.3 specific repository
baseurl=file:///aixtoolbox/RPMS/ppc-7.3/
...
```

The next step is to run the

`dnf_aixtoolbox_local.sh`

script, located in `/aixtoolbox/ezinstall/ppc`, to create local repositories from ISO media. The location of the mounted Toolbox images in Listing 4, `/aixtoolbox`, is the mountpoint for the ISO image.

The `dnf_aixtoolbox_local.sh` script does two things. First, it installs DNF if it is not already installed (note the *dnf installed successfully* message in Listing 4); second, it creates local repositories and updates the `dnf.conf` file with the repository details. (If DNF was already installed, a backup of the existing `dnf.conf` file is copied to `dnf.conf_local_bak` in `/opt/freeware/etc.`) Reviewing the resulting `dnf.conf` file (Listing 5) shows that entries were created for all three (local) AIX repositories and that `baseurl=` was set to a local file repo (i.e., the names start with `file:///`). You'll also notice that the repository names all start with `Local_`, indicating that these are local repositories.

Now when you run the command to list the available repositories (Listing 6, with verbose output), you'll notice that each repository ID starts with `Local_`, signifying that these local repositories are located on this server. Running `dnf update` updates packages to the latest version. Once the local repository server is configured from ISO media, you can now install and configure a web server and point your DNF clients to this system as a local repo server.

Configuring HTTPd

After the local repository is created, it can be shared with and accessed by DNF clients over HTTP (or HTTPS).

To do so, you first need to install and start a web server on the local repo server (e.g., Apache or NGINX). This example installs `httpd` (Apache) and a couple of tools:

```
# dnf -y install httpd
# dnf -y tcping
# dnf -y lynx
```

You would then configure the `httpd.conf` file to allow access to the location of the local repository on the repo server. I like to install the `tcping` and `lynx` tools, as well, because they are useful in monitoring and troubleshooting web servers.

The HTTP directives code (Listing 7) provides a sample of the Apache directives that can be added to the `httpd.conf` file to allow access to the repository directory. You need to restart the HTTP server for the changes to take effect. In this example, `/aixtoolbox` is the directory that contains the repository.

The `ServerName` directive was added so Apache could determine the server's fully qualified domain name during startup. The `<Directory>` directive encloses a group of directives that apply only to the named directory, sub-directories of that directory, and the files within these directories. In this case, the directory is `/aixtoolbox`.

The `AllowOverride None` directive ignores any `.htaccess` files (providing access without requiring a password). The `Require all granted` directive allows access from any IP address, and the `Alias` directive allows documents to be stored in a local filesystem other than under the default Apache `DocumentRoot`. So, for example, a request for `example.com/aixtoolbox` would cause the web server to return files in `/aixtoolbox`. Also add the stanza

```
<Location "/server-status">
    SetHandler server-status
    Require ip 127.0.0.1
</Location>
```

to the bottom section of the file to allow the status of the Apache server to be queried from the command line with the `apachectl` command. Next, start the Apache server and confirm the `httpd` daemon is responding:

```
# apachectl start
# tcping sisko 80
sisko port 80 open.
```

As shown, it should report that port 80 is open.

Checking the status of the HTTPd server shows it is healthy and ready to respond to HTTP requests (Listing 8). Now perform another test by connecting to the web server from the command line (with `lynx`), and confirm you can access the `/aixtoolbox` location over HTTP on your local repo server (Listing 9).

The output from `lynx` confirms the HTTPd server is functioning. DNF clients can now access the local repo server over HTTP.

Client Access to Local Repo Server

To configure the DNF clients to access the local repo server over HTTP, edit the `dnf.conf` file and add entries for each repository, along with the HTTP URL for the local repo server. Listing 10 shows a client `dnf.conf` file configured to access a local repo server named `sisko` over *http://*. Note that the repo IDs and names have all been set to start with `My_`, to identify a custom local configuration.

Once the `dnf.conf` file is configured, run the `dnf repolist` command to verify that all the repositories (on `sisko`) are available to the client. Listing 11 confirms that the repo IDs all start with `My_`.

Running the same command again, this time with the verbose option, shows that the local repo server is in use by the DNF client, as per the `Repo-baseurl` entry (Listing 12).

From one of your DNF clients you can now test that the local repo server works (over HTTP). In this example, you install the *zoo* package from your local repo server and turn on verbose output to see additional information, noting in which repository the *zoo*

Listing 6: Confirming Repositories Are Local

```
# dnf repolist -v | egrep "Repo-id|Repo-baseurl"
Repo-id             : Local_AIX_Toolbox
Repo-baseurl        : file:///aixtoolbox/RPMS/ppc/
Repo-id             : Local_AIX_Toolbox_73
Repo-baseurl        : file:///aixtoolbox/RPMS/ppc-7.3/
Repo-id             : Local_AIX_Toolbox_noarch
Repo-baseurl        : file:///aixtoolbox/RPMS/noarch/
```

Listing 7: Apache Directives

```
ServerName sisko.edu.ibm
...
<Directory "/aixtoolbox">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Require all granted
</Directory>
Alias /aixtoolbox "/aixtoolbox"
```

Listing 8: Apache Server Status

```
# apachectl status
   Apache Server Status for localhost (via 127.0.0.1)
   Server Version: Apache/2.4.58 (Unix)
   ...etc...
```

Listing 9: Confirming Access over HTTP

```
# lynx http://sisko//aixtoolbox -dump
     Index of /aixtoolbox

     * [1]Parent Directory
     * [2].Version
     * [3]CONTENTS.toolbox
     * [4]LICENSES/
     * [5]README.toolbox
     * [6]RPMS.bygroup/
     * [7]RPMS/
     * [8]SRPMS/
     * [9]contrib/
     * [10]docs/
     * [11]ezinstall/
     * [12]installp/
     * [13]usr/
...etc...
```

Listing 10: Configuring DNF Clients

```
[My_AIX_Toolbox]
name=My AIX generic repository
baseurl=http://sisko:80//aixtoolbox/RPMS/ppc
enabled=1
gpgcheck=0

[My_AIX_Toolbox_noarch]
name=My AIX noarch repository
baseurl=http://sisko:80//aixtoolbox/RPMS/noarch/
enabled=1
gpgcheck=0

[My_AIX_Toolbox_73]
name=My AIX 7.3 specific repository
baseurl=http://sisko:80//aixtoolbox/RPMS/ppc-7.3/
enabled=1
gpgcheck=0
```

Listing 11: Checking Repo IDs

```
# dnf repolist
repo id                     repo name
My_AIX_Toolbox              AIX generic repository
My_AIX_Toolbox_73           AIX 7.3 specific repository
My_AIX_Toolbox_noarch       AIX noarch repository
```

Listing 12: Local Repo Server In Use

```
# dnf repolist -v | grep Repo-baseurl
Repo-baseurl       : http://sisko:80//aixtoolbox/RPMS/ppc
Repo-baseurl       : http://sisko:80//aixtoolbox/RPMS/ppc-7.3/
Repo-baseurl       : http://sisko:80//aixtoolbox/RPMS/noarch/
```

Listing 13: Installing zoo

```
# dnf -yv install zoo
...etc...
--> Starting dependency resolution
---> Package zoo.ppc 2.10-5 will be installed
...etc..
Installing:
 zoo       ppc       2.10-5           My_AIX_Toolbox              62 k
...etc...

Installed:
  zoo-2.10-5.ppc

Complete!
```

Listing 14: Find Package in Logfile

```
...
2024-10-11T04:51:46Z DEBUG DNF version: 4.2.17
2024-10-11T04:51:46Z DDEBUG Command: dnf -yv install zoo
...
2024-11-01T01:15:33Z DEBUG ---> Package zoo.ppc 2.10-5 will be installed
...etc...
Installing:
 zoo.......ppc.......2.10-5..........My_AIX_Toolbox..............62 k

...etc...
Installed:
  zoo-2.10-5.ppc
       ...
```

package is installed (i.e., from `My_AIX_Toolbox`; **Listing 13**).

Back on the repo server, you can search the web server logfile, `access_log`, for the HTTP `GET` request for the zoo RPM file from the client:

```
# grep zoo /var/log/httpd/access_log
10.8.12.141 - - [➘
  11/Oct/2024:00:14:14 -0500] ➘
  "GET //aixtoolbox/RPMS/ppc/zoo-2.10-5.➘
   aix5.1.ppc.rpm HTTP/1.1" 200 63688
```

On the DNF client, you can open the `dnf.log` file with view `/var/log/dnf.log` and search the log for the installation of the *zoo* package with DNF (**Listing 14**).

With these steps complete, you've now configured your AIX DNF client to use a local repo server. Next, I'll review how you can use DNF to manage OSS packages on AIX.

Open Source Packages on AIX

Now that you have DNF installed and a local repo server up and running, you'll want to look at how to install, make operational, and manage OSS packages on AIX.

As an example we'll install the open source MariaDB Server from the AIX Toolbox. This popular open source relational database was developed by the original developers of MySQL **[10]**.

From the AIX Toolbox, use DNF to install the MariaDB Server:

```
dnf -y install mariadb-server
```

To avoid any out-of-memory messages when starting the database, change the AIX user limits for the *root* and *mysql* users, with the chuser command (**Listing 15**).

You can verify that the user limits have been changed successfully with the

```
lsuser -f <username>
```

command and check that the updated attributes are set to -1.

Now you can start a new root user session with `su -`, which starts a new root shell with the updated user attributes from the previous step. You can check that the new limits for the root user have taken effect by running the `ulimit` command before and after the `su -` command. Note that the majority of resource limits are now set to `unlimited` (**Listing 16**).

The next step is to prepare the database server setup to handle the initialization tasks that must be performed before the database server (`mysqld`) is ready to use. It initializes the database data directory and creates the system tables it contains, initializes the system tablespace and related data structures needed to manage InnoDB tables, loads the server-side help tables, installs the sys schema, and creates an administrative account (**Listing 17**).

Now you can start the database server with the command:

```
# echo "/opt/freeware/bin/mysqld_safe ➘
  --datadir='/opt/freeware/var/lib/mysql/➘
  data'" | at now
```

Next, check that the MariaDB server process is running (**Listing 18**).

You can check the status of the server and get a prompt with:

Listing 15: Changing Limits

```
# chuser core=-1 cpu=-1 data=-1 fsize=-1 stack=-1 nofiles=-1 core_hard=-1 cpu_hard=-1 data_hard=-1 fsize_hard=-1 stack_hard=-1 nofiles_hard=-1 root
# chuser core=-1 cpu=-1 data=-1 fsize=-1 stack=-1 nofiles=-1 core_hard=-1 cpu_hard=-1 data_hard=-1 fsize_hard=-1 stack_hard=-1 nofiles_hard=-1 mysql
```

Listing 16: Checking New Limits

```
# ulimit -a
time(seconds)         unlimited
file(blocks)          2097151
data(kbytes)          131072
stack(kbytes)         32768
memory(kbytes)        32768
coredump(blocks)      2097151
nofiles(descriptors)  2000
threads(per process)  unlimited
processes(per user)   128

# su -

# ulimit -a
time(seconds)         unlimited
file(blocks)          unlimited
data(kbytes)          unlimited
stack(kbytes)         unlimited
memory(kbytes)        32768
coredump(blocks)      unlimited
nofiles(descriptors)  unlimited
threads(per process)  unlimited
processes(per user)   128
```

```
/opt/freeware/bin/mysql -u root mysql
...
MariaDB [mysql]>
```

At the resulting prompt, type `status` to see the active status of the database server (**Listing 19**); then, type `exit;` to return to the shell, change to the *mysql* user, and connect to the database server (**Listing 20**).

At the prompt on the last line, create a database named MARIAIX:

```
CREATE DATABASE MARIAIX;
```

To confirm the database was created successfully, list the databases, with `SHOW DATABASES;`. Type `exit;` to return to the shell prompt, and type `exit` (without the semicolon) to return to the root shell prompt (**Listing 21**).

You can stop the database server with the `mysqladmin` tool:

```
mysqladmin -u root ↩
  -p shutdown
```

Enter the root password when prompted. However, MariaDB Server is also integrated into the system resource

Listing 17: Initializing Database Server

```
# mysql_install_db --user=mysql
Installing MariaDB/MySQL system tables in '/opt/freeware/var/lib/mysql/
   data' ...
OK

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system


Two all-privilege accounts were created.
One is root@localhost, it has no password, but you need to
be system 'root' user to connect. Use, for example, sudo mysql
The second is mysql@localhost, it has no password either, but
you need to be the system 'mysql' user to connect.
After connecting you can set the password, if you would need to be able
   to connect as any of these users with a password and without sudo

See the MariaDB Knowledgebase at http://mariadb.com/kb

You can start the MariaDB daemon with:
cd '/opt/freeware' ; /opt/freeware/bin/mysqld_safe --datadir='/opt/
   freeware/var/lib/mysql/data'

You can test the MariaDB daemon with mysql-test-run.pl
cd '/opt/freeware/mysql-test' ; perl mysql-test-run.pl

Please report any problems at http://mariadb.org/jira

The latest information about MariaDB is available at http://mariadb.
   org/.

Consider joining MariaDB's strong and vibrant community:
https://mariadb.org/get-involved/
#
```

Listing 18: Checking `mysqld`

```
# ps -ef | grep mysqld
   mysql  8323484 18219314   0 21:39:57  pts/1  0:00 /opt/freeware/libexec/
      mysqld --basedir=/opt/freeware --datadir=/opt/freeware/var/lib/mysql/data
      --plugin-dir=/opt/freeware/lib64/mariadb/plugin --user=mysql --log-error=/
      opt/freeware/var/log/mysql/mysqld.log --pid-file=/opt/freeware/var/lib/
      mysql/mysqld.pid --socket=/opt/freeware/var/lib/mysql/mysql.sock
    root 18219314 14418290   0 21:39:57  pts/1  0:00 /bin/sh /opt/freeware/bin/
mysqld_safe
```

Listing 19: MariaDB Server Status

```
MariaDB [mysql]> status
--------------
/opt/freeware/bin/mysql  Ver 15.1 Distrib 10.4.26-MariaDB, for AIX7.1 (powerpc)
using readline 5.1

Connection id:          9
Current database:       mysql
Current user:           root@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server:                 MariaDB
Server version:         10.4.26-MariaDB Source distribution
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    latin1
Conn.  characterset:    latin1
UNIX socket:            /opt/freeware/var/lib/mysql/mysql.sock
Uptime:                 17 sec

Threads: 6  Questions: 75  Slow queries: 0  Opens: 37  Flush tables: 1  Open
tables: 30  Queries per second avg: 0.013
--------------
```

Listing 20: Connect User to Database Server

```
$ su - mysql
$ /opt/freeware/bin/mysql -u mysql mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.26-MariaDB Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

readline: warning: turning on OPOST for terminal
readline: warning: turning on OPOST for terminal
MariaDB [mysql]>
```

controller (SRC) framework when installed on AIX, meaning you can stop and start it with the built-in AIX commands. You can confirm the *mariadb* SRC subsystem exists on the system with the command in Listing 22.

Listing 21: Displaying Database

```
MariaDB [mysql]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| MARIAIX            |
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
5 rows in set (0.000 sec)
MariaDB [mysql]> exit;
Bye
$ exit
#
```

Listing 22: MariaDB SRC Subsystem Entry

```
# lssrc -Ss mariadb
#subsysname:synonym:cmdargs:path:uid:auditid:standi
   n:standout:standerr:action:multi:contact:svrkey
   :svrmtype:priority:signorm:sigforce:display:wait
   time:grpname:mariadb::-c "ulimit -n unlimited; ulimit
   -c 10; ulimit -m unlimited; ulimit -d unlimited; /
   opt/freeware/libexec/mysqld" :/usr/bin/sh:27:0:/dev/
   console:/dev/console:/dev/console:-O:-Q:-S:0:0:20:
   15:9:-d:20::
```

Listing 23: Starting MariaDB with AIX Command

```
# startsrc -s mariadb
0513-059 The mariadb Subsystem has been started.
   Subsystem PID is 5636486.

# lssrc -s mariadb
Subsystem         Group            PID          Status
 mariadb                           5636486      active

# ps -fu mysql
     UID      PID     PPID    C    STIME    TTY  TIME CMD
   mysql  5636486  4129208    0 19:39:27      -  0:00 /
      opt/freeware/libexec/mysqld
```

Listing 24: Stopping MariaDB with AIX command

```
# stopsrc -s mariadb
0513-044 The mariadb Subsystem was requested to stop.

# lssrc -s mariadb
Subsystem        Group            PID         Status
 mariadb                                      inoperative
```

Now start the database server from the AIX SRC and confirm the *mariadb* subsystem is now active (Listing 23). You can stop the DB server and confirm the *mariadb* subsystem is *inoperative* (Listing 24). Congratulations, you have just used DNF to install your first open source database application and create and manage a MariaDB database on AIX.

Getting Help

RPM packages from the AIX Toolbox are not supported through traditional IBM AIX support. However, assistance is available through the IBM AIX Open Source Community Discussion forum, which can lend assistance for such things as installing or configuring packages, possible package defects, understanding error messages from attempts to install or configure a package, security vulnerabilities found in a particular RPM, and dependency issues with a package from the Toolbox site. A number of blog posts and videos are also useful [11]-[14]. Visit the forum [15] and the official documentation for the Toolbox [16] to learn more. The site provides vital information relating to the Toolbox. The website has several tabs at the top of the page (Figure 6), which I encourage you to review at your leisure.

If you're interested in some best practices advice on planning and maintaining the AIX Toolbox packages [17] or if you're looking for recommendations on AIX `$LIBPATH` settings with Toolbox packages [18], I highly recommend you review the links in the Info section. Also, if you're in need of a solution that removes all the installed RPMs from your AIX system so you can perform a clean installation of DNF (which can sometimes be required on decades old AIX systems that have been gradually migrated to more recent versions of AIX), again, check the links for some tips and tricks – in particular, the `destroyRPMS` script [19]; however, make a backup first, please!.

Conclusion

My hope is that this article, although introductory, has shown you how to embrace the power and flexibility of open source software across your entire AIX landscape. IBM is invested in the AIX Toolbox, with dedicated developers working to create and update various OSS packages that are freely available to everyone. To learn more, take a look at the online self-paced virtual course [20]. This class will teach you how to install and manage open source software on AIX (with DNF) and give you hands-on access to a real AIX lab environment where you can implement what you've learned. ■

Info

[1] "Building Open Source RPM Packages on IBM AIX" by Sanket Rathi, David Clissold, Nitish Mishra, and Sangamesh Mallayya, *IBM Developer*, June 2017: [https://developer.ibm.com/articles/au-aix-build-open-source-rpm-packages/]

[2] AIX Toolbox: [https://www.ibm.com/support/pages/aix-toolbox-open-source-software-overview]

[3] Dependency hell: [https://en.wikipedia.org/wiki/Dependency_hell]

[4] DNF install script (download): [https://public.dhe.ibm.com/aix/freeSoftware/aixtoolbox/ezinstall/ppc/dnf_aixtoolbox.sh]

[5] Install script flags reference: [https://community.ibm.com/community/user/power/blogs/sangamesh-mallayya1/2021/05/28/dnf-is-now-available-on-aix-toolbox]

[6] AIX PPC repository: [https://public.dhe.ibm.com/aix/freeSoftware/aixtoolbox/ezinstall/ppc/]

[7] "Configuring YUM and Creating Local Repositories on IBM AIX" by Sangamesh Mallayya, Nitish Mishra, Sanket Rathi, and Ayappan P, *IBM Developer*, October 2018: [https://developer.ibm.com/articles/configure-yum-on-aix/]

[8] ESS: [http://www.ibm.com/servers/eserver/ess/index.wss]

[9] Passport Advantage Online (PAO): [https://www.ibm.com/software/passportadvantage/pao_customer.html]

[10] MariaDB Server: [https://mariadb.org/about/]

[11] "Configuring DNF and Creating Local Repositories on IBM AIX" by Sangamesh Mallayya, Sanket Rathi, Ayappan P, Harshith K, *IBM Developer*, September 2023: [https://developer.ibm.com/tutorials/awb-configuring-dnf-create-local-repos-ibm-aix/]

[12] "Configuring DNF Local Repositories Containing AIX Toolbox RPM Packages" by Sangamesh Mallayya, *IBM TechXchange*, June 2024: [https://ibm.biz/BdaZZ6]

[13] "Creating Local Repo with DNF and AIX Toolbox Media Image" by Sangamesh Mallayya, *IBM TechXchange*, February 2022: [https://community.ibm.com/community/user/power/blogs/sangamesh-mallayya1/2022/02/09/creating-local-repo-with-dnf-and-aix-toolbox-media]

[14] "Configuring DNF local repositories using AIX Toolbox Media Image" by Sangamesh Mallayya, *IBM TechXchange*, June 2024: [https://ibm.biz/BdaZZt]

[15] AIX open source community discussion forum: [https://community.ibm.com/community/user/power/communities/community-home?CommunityKey=10c1d831-47ee-4d92-a138-b03f7896f7c9]

[16] AIX Toolbox for open source software: [https://www.ibm.com/support/pages/node/882892]

[17] "Planning and Maintaining AIX Toolbox Packages" by Jan Harris, *IBM TechXchange*, October 2024: [https://community.ibm.com/community/user/power/blogs/jan-harris1/2024/10/17/planning-and-maintaining-aix-toolbox-packages]

[18] AIX LIBPATH recommendations: [https://www.ibm.com/support/pages/node/7091281]

[19] Remove all RPMs from system and install DNF (download): [https://public.dhe.ibm.com/aix/freeSoftware/aixtoolbox/contrib/destroyRPMS]

[20] Open Source Software on IBM AIX (IBM training course code AN35DG): [https://www.ibm.com/training/course/open-source-software-on-ibm-aix-AN35DG]

Author

Chris Gibson is a technical specialist for the IBM Power Technical Training team. He creates technical training content for IBM Power, AIX, and PowerVM. Over his 25 years of working with AIX (as an administrator, consultant, instructor, presenter, speaker, blogger, and content creator), Chris has written and published many technical articles and has co-authored several IBM Redbooks on IBM AIX and Power. Chris resides in Melbourne, Australia. You can stay in touch by following his blog at [http://gibsonnet.net/blog/cgaix/] or from his Linkedin account, [https://www.linkedin.com/in/chris-gibson-9921253/].

AIX Toolbox for Open Source Software

GNU and open source tools for AIX

Overview
License
Downloads alphabetical
Downloads by date
Get Started
Get Help
What's New

Figure 6: GNU and open source tools for AIX.

Risk mitigation for Active Directory

Deviations

Active Directory default settings could expose your AD environment to security vulnerabilities. We look at account hygiene and risk mitigation in authentication, login data, PKI, domain join, and more.

By Evgenij Smirnov

In a previous article [1], I analyzed the problems that have given Active Directory (AD) a poor reputation among security admins. However, only a minor share of known vulnerabilities originates from the nature of AD itself. Many problems are the result of the default settings that Microsoft still delivers with AD; configurations and processes that admin teams implement in their AD organizations can be even more problematic.

To design and operate AD as securely as possible – in the context of this article I talk about Active Directory Domain Services (AD DS) – it is very important to understand which of today's cyberthreats affect AD, what goals attackers pursue, and what techniques they deploy to achieve their goals. The current talk can give non-experts the impression that an IT infrastructure is virtually defenseless against cybercriminals if it is based on AD. In reality, though, AD is never the first point of contact or the gateway for the attacker, and hijacking AD is just as rarely the ultimate target.

The initial contact is usually made over a user device (phishing, drive-by downloads) or an application (Exchange administrators might well remember the HAFNIUM zero-day exploit). The attackers' objectives range from total destruction (wiper) to the theft of very specific information (research data for industrial espionage, commercial documents for market manipulation, or whale phishing attacks on high-level executives). Somewhere in between you also see attackers encrypting an organization's entire data as a means of blackmail (ransomware), injecting code into the development cycle (software supply chain attacks), or influencing the networked physical infrastructure (power generation or gas pipeline attacks).

Mitigating Risks

In most AD implementations that have grown historically, attackers find numerous opportunities to achieve their intended targets relatively quickly without attracting attention at an early stage. For initial reconnaissance, a hijacked user account, workstation, or application server can help the attacker discover extensive details about the environment (configurations, permissions, escalation paths) using the operating system's on-board resources. Sometimes access credentials for other accounts are already known – for example, local accounts created by group policies or role accounts with passwords in the description field. I already talked about this part of the attack chain and methods of restricting the visibility of highly privileged objects in the first article [1].

Once the attacker succeeds in gaining control over one or more member computers and is able to execute arbitrary processes in the system context of these machines, they are potentially able to access the credentials of other accounts that have logged on to these computers. In most cases, the material captured is only password hashes, but as long as the NT LAN Manager (NTLM) authentication protocol and the RC4 stream cipher are supported in the environment, a hash can be used for authentication (pass-the-hash attack)

Photo by cdd20 on Unsplash

or at least to request Kerberos tickets (overpass-the-hash attack).

Hashes that use these methods can be cracked with reasonable effort if the original password is relatively short and simple. If the captured hash material contains the access credentials of highly privileged accounts, the attacker can leverage these credentials to create new privileged identities, extend replication permissions, or manipulate audit settings. If legacy service accounts are used in the environment, they can fall victim to Kerberoasting. These service accounts are often given a non-expiring but relatively short password, which makes cracking a service ticket possible with reasonable overhead.

If the attacker comes across replication permissions during the reconnaissance phase and can exploit them directly, they can replicate valuable data from AD, including the password hashes of the Kerberos ticket-granting ticket (KRBTGT) account, which is the ultimate authority for Kerberos authentication in AD domains (Golden Ticket attack), and Data Protection Application Programming Interface (DPAPI) backup keys, which can be used to decrypt all member computer and AD user secrets protected by DPAPI.

If an enterprise public key infrastructure (PKI) is implemented in the AD forest and is not sufficiently hardened, the bad guys can have Kerberos certificates issued that let them log in directly in the context of highly privileged accounts. Computer and service accounts enable further attacks on access data by exploiting misconfigurations in the Kerberos delegation.

Once the attacker is able to manipulate group policies, they usually get far closer to their ultimate goal. In ransomware or wiper scenarios, for example, the malware can be rolled out to the entire environment by startup scripts or scheduled tasks and detonated at the intended time. However, persistence and escalation are also possible; if the intruder manages to link a group policy to the organizational unit (OU) of the domain controller (DC) that assigns Local Admins rights to user accounts on member PCs, they can convert these accounts into Domain Admins.

These and a few other related techniques must be the focus of any initiative aiming to make an AD environment as secure as possible. Bear in mind that IT security is a risk management discipline; it's not about risk avoidance, it's about risk mitigation. If you manage to make every potential attack path as complex and expensive as possible with your hardening measures, cybercriminals may turn to easier prey.

Regulating Authentication

In terms of risk management, a secure AD design reduces the probability of login data misuse; the more extensive the privileges of the target identity, the greater the risk. Additionally, the effect of credential theft must be kept to a minimum, especially if this theft, including clear text passwords, cannot be ruled out, as is the case with domain join accounts, for example.

Limiting the damage requires rigorous implementation of the principle of least privileges, which starts with restricting read and write permissions in AD itself only to what is absolutely necessary for the purpose of the account in question. However, even the ability to log on to computers and start processes involves authorization that can be abused. Under certain circumstances, even simple read-only network access to "officially" shared resources is a valuable asset from the attacker's point of view (e.g., if the target of the attack is stealing a specific document).

Designing a full set of access rights for a least privileges model is a major undertaking, and very few administrators and system architects are capable of creating this kind of design at the drawing board. The task is also complicated by the fact that many of the permissions that ideally need to be revoked are assigned by default to virtual groups such as Authenticated Users or Everyone, from which you cannot remove individual users or computers.

Therefore, you need to pluck up courage and boldly remove the authorization entries of these virtual groups, replacing them with a combination of options that delegate the required permissions to groups that actually exist. This change gives you the opportunity to make highly privileged objects invisible to normal users and computers from the outset. You will always want to use scripts to assign permissions in AD, because it already provides 90 percent of the code you need to check later for correct permissions and helps you assign them quickly in another forest.

Protecting Login Data

When it comes to protecting credentials against theft and misuse, you need to consider the various threat scenarios. First is the disclosure of the username and password (i.e., the famous Post-it note under the keyboard or the `passwords.txt` file on the desktop). In the modern cloud world, you can at least mitigate these threats to some extent with multifactor authentication (MFA). Traditional on-premises technologies offer far less leeway in this respect, which leaves consistent training of all employees, including admins, as the only way to counteract physical password sharing. Another approach that also works well locally is to dispense with passwords in favor of smart cards or Windows Hello for Business. To begin, check whether all applications in your infrastructure support Kerberos or at least NTLM. If you need to run an important web application used by many that insists on users entering usernames and passwords on a login screen, make sure you check whether it supports Active Directory Federation Services (ADFS) as an authentication method.

The next item on the list is brute-force attacks on explicit authentication. The obvious answer to this type of threat is an account lockout policy, but automatic account lockout is a double-edged sword – after all, it can paralyze the entire operation with minimal effort. If you really want to use this policy, set the number of failed logins until lockout to a far higher value than the *10* that both Microsoft and the Center for Internet Security (CIS) recommend by default. One possible solution without denial of service potential is adaptive throttling. Applications with a web front end regularly use this technology, as any Fritz!Box user who has mistyped their password knows. Microsoft introduced the authentication rate limiter for Server Message Block (SMB) file shares in Server 2025. Although this method is not adaptive in the sense that the interval grows with each additional unsuccessful login attempt, it is definitely a step in the right direction.

Authentication policies can, to a certain extent, help limit the attack surface for brute force. For example, a Tier 0 account cannot be blocked by a brute-force attack on application servers, a service account cannot be blocked by a client, and a normal user cannot be blocked by attacks on the NETLOGON share of a DC. As a final means of protection, you can always monitor failed login attempts – of course, you only achieve advantages in operations if you actually follow up on the messages without delay.

Brute-force attacks on captured password hashes include hashes that are read from logon sessions on workstations or servers and techniques such as Kerberoasting, which is based on the data structure of a Kerberos service ticket and takes advantage of the ability of every user to request a service ticket for every service by default.

The recommended approach is to define rigorously where users can log in, for which you can use authentication and group policies (assigning user rights), although the former is far more effective. This approach is known as administrative tiering for the protection of highly privileged accounts, but you can also use it within a tier (zoning) to channel potentially accessible credentials further and make both escalation and lateral movement more difficult. Microsoft also offers Credential Guard on all supported Windows versions, which is an extremely effective virtualization-based technology you need to enable.

The only thing that really helps to make the last part of the attack more difficult – that is, reconstructing the plain-text password from the hash or service ticket by brute force – is a drastic increase in password length, ideally enforced by password policies or by the process you set up for changing passwords. User education with the goal of replacing passwords with passphrases and therefore implementing long but memorable passwords is a very worthwhile endeavor.

Recommendations from both the US National Institute of Standards and Technology (NIST) and German Federal Office for Information Security (BSI) even go so far as to recommend that companies do away with regular password changes for passwords greater than a certain minimum length (typically 20 characters). In a hypothetical environment where a password is typed in a dialog from a keyboard, this recommendation might even be justified. Unfortunately, password hashes are often still valuable assets without brute force, which forces admins to implement a workable password change procedure for every kind of account – normal users, admins, service accounts, and even break glass (emergency access) accounts.

When attackers inject password hashes directly (pass-the-hash or overpass-the-hash), they are exploiting two peculiarities in the Windows authentication system. First, NTLM only requires the password hash for the entire authentication process, and second, the RC4 hash in Kerberos is identical to the NTLM hash. If the hashes are injected directly, the length of the underlying passwords is irrelevant. As long as your environment still accepts NTLM or Kerberos allows RC4 cryptography, abolishing regular password changes is not an option; regular password changes are vital for security.

You will want to follow the authentication guidelines or at least use the Protected Users group for privileged accounts so that NTLM and RC4 are ruled out or are restricted to specific communication relationships. Also, consider switching to Group Managed Service Accounts (gMSAs) for service and task accounts if you have not already done so. AD automatically carries out password changes in this case. All the other approaches described previously that prevent hashes from being captured can also help shut off this attack route.

Of course, sniffing hashes from login sessions is not the only way to obtain this valuable material. A whole bunch of relay and replay attack techniques, all of which have the word "potato" in their names **[2]**, aim to collect hash material. Once attackers have managed to grab replication authorizations, they can read the hashes directly from AD by directory replication. Of course, an unprotected system state backup of a DC, should it fall into the wrong hands, offers unhindered access to this information. The statement that Microsoft only supports a system state backup in the event of a disaster is one of the most persistent myths about disaster recovery in the AD ecosystem. System state backups of DCs can be found in many environments, and if you base your disaster recovery strategy for AD on them, you need at least to ensure that access to these datasets is extremely restrictive.

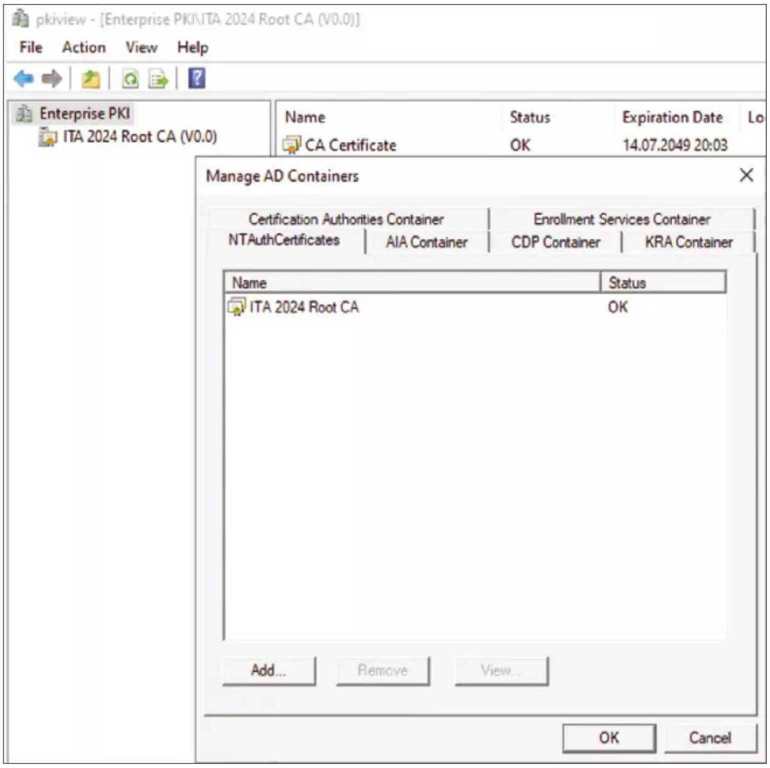
Trust Issues

In addition to the well-known dangerous misconfigurations of AD, a number of attack vectors have begun to be addressed in recent years that target AD DS but use a different AD – AD Certificate Services (AD CS; Microsoft's implementation of an enterprise PKI). The first systematic discussion of AD CS vulnerabilities that lead to the compromise of AD DS is mentioned in an article **[3]** that references a whitepaper by Will Schroeder and Lee Christensen from 2021. The attack techniques described there have since been refined and extended multiple times and should make clear to every security admin the serious potential for trouble. Before you decide on the optimal PKI strategy for your organization, you need to be aware of the PKI-based attack vectors and the techniques you can use to mitigate the effect of enterprise PKI on your AD security.

Every PKI is based on trust relationships. Threats from the certificate services are based on two ways of abusing these relationships: (1) Kerberos in Windows accepting certificates as a valid authentication method, and (2) certificate types not directly ensuring successful Kerberos authentication, but still enabling communication relationships. Both smart cards and Windows Hello are based on the first mechanism, and the sub-protocol used is Public Key Cryptography for Initial Authentication (PKINIT).

That said, the Key Distribution Center (KDC) on a DC does not trust every certificate authority (CA) that can be traced back to the trusted roots of the DC for the validation of smart card certificates. Instead, it uses a special NTAuth certificate store (Figure 1). An enterprise CA automatically registers in the store at install time, irrespective of whether or not the organization intends to issue authentication-capable certificates from this PKI. In other words, attacks on a root CA can point an attacker to certificates that are valid for Kerberos and issued to highly privileged identities such as admin accounts, DCs, or even entire domains (which means that even unidirectional trust relationships can be compromised).

The second type of threat is not as serious but is more difficult to handle. These scenarios primarily involve server certificates that allow an attacker to spoof an HTTPS page without leading to certificate messages in the applications. Of course, computer certificates that are valid for 802.1X authentication can also give the attacker access to networks on which the targets reside.



Name	Status	Expiration Date	Lo
CA Certificate	OK	14.07.2049 20:03	

Name	Status
ITA 2024 Root CA	OK

Figure 1: Kerberos only accepts certificates for authentication whose issuing CA can be found in the NTAuth container.

Both problems are exacerbated because the Microsoft PKI makes it very easy to create a situation in which a user who is not particularly highly privileged can request and receive certificate types that are of interest to attackers and can also freely define the subject name of these certificates. This scenario would enable the attacker to issue a smart card certificate for the domain admin and proceed to hijack the domain. If the attacker's target lies behind a web portal that requires the password of an authorized user for entry, the attacker could have an HTTPS certificate issued for the corresponding URL and then spoof DNS for direct user access to a web server that they control.

Operating PKI the Right Way

For a secure PKI design (which can also involve getting rid of your internal PKI), defining who can apply for certificates, which procedures they can use to do so, and to what extent the applicants can be trusted is important:

- If you do not use certificate-based authentication in Kerberos (strongly recommended), remove your PKI's certificates from the NTAuth container. You can add the certificates again later if needed.
- Treat PKI as a Tier 0 system and regulate admin access accordingly. Incidentally, this applies regardless of whether you use the Microsoft PKI (AD CS) or another product.
- In a multilevel PKI, all CAs need to be assigned to Tier 0.
- Use CA restrictions to channel how certificates are issued. If required, provide separate CAs for smart card certificates and other types, such as web server certificates.
- Operating multiple PKI trees with separate root CAs for different applications is also a useful option.
- Do not install PKI web services just because you think you might need them. If you do need them, configure them in a way that is fit for their purposes. Nobody should need the outdated Certificate Services Web

Enrollment service today because it is based on ActiveX controls.

- Only Tier 0 administrators should be able to request smart card certificates. If a smart card is required for lower tier admins or normal users, use an enrollment agent instead of giving users permission to request these certificates. If you have no other option, make sure an administrator has to approve the smart card certificates, regardless of who the original requester is.
- Be sure to use the *TameMyCerts* policy module with your Microsoft PKI, so you will be able to define binding rules on the addresses and identities that your certificates can contain.
- Use deterministic Online Certificate Status Protocol (OCSP) configurations [4] instead of a simple certificate revocation list (CRL), at least for highly sensitive certificates. In this way, certificates issued by the attacker from within the PKI can still be valid, but you can at least catch the case in which the attacker has obtained the private key of an issuing CA and signs certificates without the PKI.

If your organization does not yet have a clearly formulated strategy for dealing with certificates, a safer alternative is not to operate an internal PKI and to purchase HTTPS certificates for web applications from a public certification authority. You will need either a publicly resolvable domain name (which also belongs to you) or a split-horizon DNS configuration that resolves public names to private IP addresses when internal DNS servers are queried.

Manage Configurations Cleanly

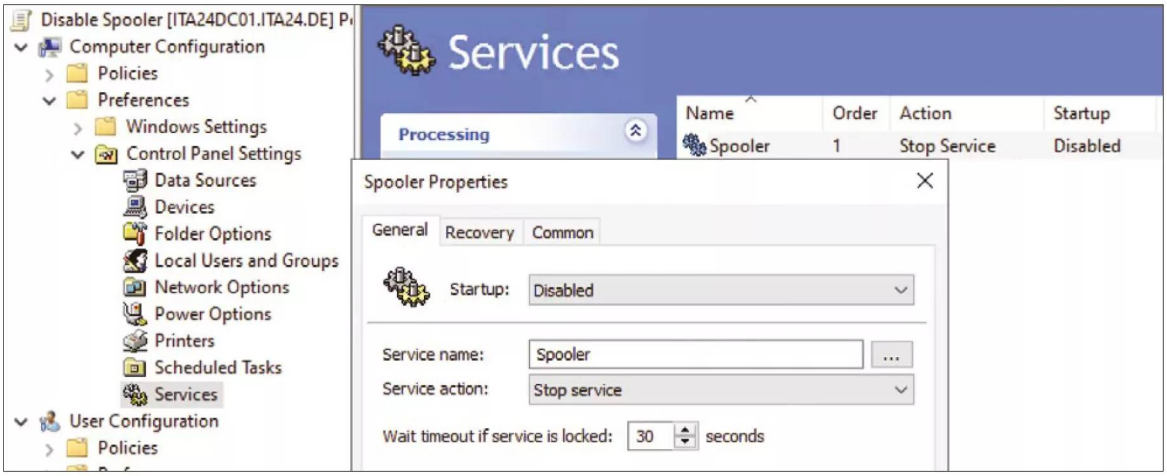
Many of the measures that will make your AD significantly more secure are not rooted in AD itself, but in the infrastructure that surrounds it, including Windows servers with the DC role. Because the attacker's first contact with the environment is either via an endpoint or an externally published application server, you can already significantly reduce your environment's attack surface by deploying the following on these machines:

- Up-to-date, managed and intelligent anti-malware
- Process whitelisting – ideally Windows Defender Application Control or at least AppLocker
- Restricted administrative permissions to prevent regular interactive logins with admin privileges and Internet access from within administrative sessions
- Credential Guard

Firewalls help make lateral movement and escalation more difficult. You also need to configure them with a whitelist – only what is explicitly permitted is possible. You can set up initial obstacles with some very simple firewall rules such that:

- Clients are not allowed to communicate with each other, irrespective of the protocol, unless you are one of the few organizations that have integrated Windows Update Delivery Optimization into your management. In combination with the Local Administrator Password Solution (LAPS), this setup virtually eliminates lateral movement.
- Access to management networks for hardware and virtualization management, network devices, storage, and climate and power control is only permitted from selected devices, but never from client or printer networks. If you want to be on the safe side, set up a jump host in the network segments and secure access to it with MFA – in the simplest case with a remote desktop gateway.
- Management protocols (e.g., Windows Remote Management, WinRM, and Windows Management Instrumentation,WMI, and now increasingly also SSH on infrastructure servers) are only accessible from a small number of explicitly specified management and monitoring machines.

When choosing monitoring and management applications, rely on agent-based products whose agents establish connections from the managed devices to the management system



Name	Order	Action	Startup
Spooler	1	Stop Service	Disabled

Startup: Disabled

Service name: Spooler

Service action: Stop service

Wait timeout if service is locked: 30 seconds

Figure 2: Group Policy Preferences is the best option for deactivating services.

and not vice versa. The fewer incoming communication relationships required for your systems to work correctly, the better.

Define binding rules on who can edit group policies and, above all, who can link them to OUs with highly privileged objects (DCs, Tier 0 admins), domains, and AD sites. Unfortunately, the Group Policy Management console that comes with AD is not very cooperative if you try to restrict the permissions too far. Consider either placing all group policy management in Tier 0 or using a third-party product for Group Policy Object (GPO) management that allows effective delegation.

Use BitLocker on DCs wherever you have a serious risk of hard disks being stolen – whether a physical server or a virtual disk image on centralized storage. Of course, BitLocker only has the potential to increase the security of your DCs if the DC is stolen in this way and does not automatically unlock itself when booting up, which in your day-to-day admin work, rules out unattended restarts of your DCs. You might want to consider using the BitLocker Network Unlock key protector for this reason.

Disable all unnecessary services on highly privileged systems. Unfortunately, Microsoft no longer provides its own recommendations and has not done so since Server 2019; therefore, you have to rely on your own experience, information from the community, and trial and error. In any case, you will want to deactivate the print spooler services on your Tier 0 systems permanently (Figure 2).

If you set up shared printers in your AD, you will find that the print queue objects are then no longer deleted automatically. However, this is a very small price to pay for an important security feature.

Even after you have created and rolled out a configuration for both your AD forest and the machines in it, your work is not done. From now on you need to monitor the status continuously to detect and remedy deviations from the defined configuration (configuration drift) at an early stage. Depending on which AD and configuration management technologies are established in your organization, you can rely on monitoring, Desired State Configuration, infrastructure-as-code, or other techniques. The important thing is that you identify deviations and react to them.

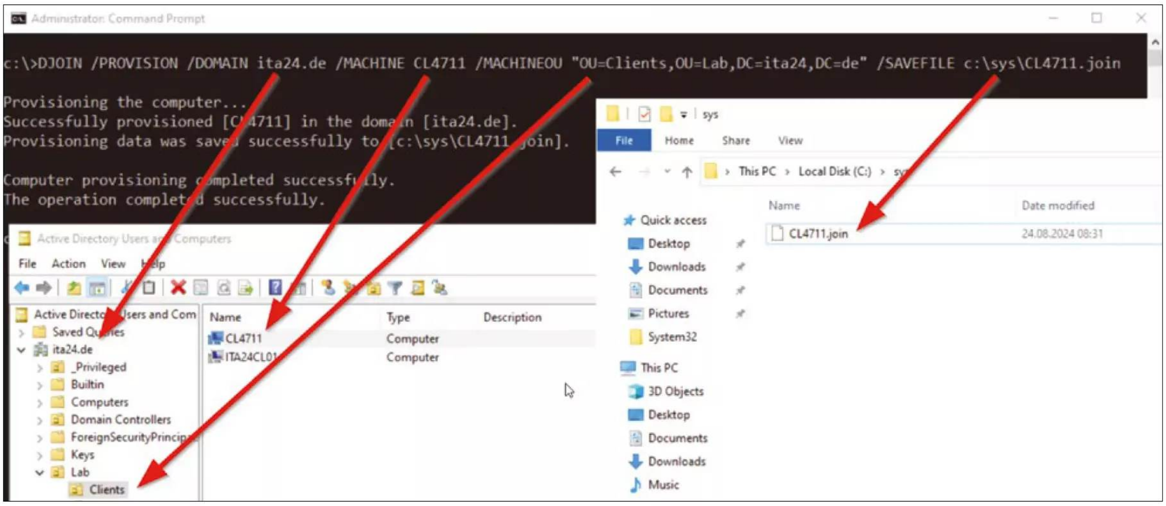
Domain Join as a Security Risk

One process that exposes many potential attack vectors is adding a new computer to the domain. As part of general account hygiene, you will want to remove the permission for adding up to 10 computers to the domain, which is assigned to each user by default. This action will at least prevent a potential attacker from creating new computer objects very early on in their attack chain and adding machines of their choice to your AD.

The domain join process is far more serious for servers. If a domain admin initiates this process manually on the new machine, the owner of the resulting computer object is the Domain Admins group, which is a good and secure choice. However, if you want to automate the process, you need to be aware that the credentials of the account intended for the domain join are effectively available in the clear, because these credentials need to be transmitted to a freshly installed computer that does not have a trust relationship. Therefore, you must strive to limit the rights of the join user to the necessary minimum – with the result that the join user will own the computer object in AD. If you are not careful and the new server becomes a Tier 0 system, perhaps even a DC, you'll have created a highly privileged system whose computer object belongs to a user whose credentials are effectively exposed in plain text.

To get out of this dilemma, several methods can secure (even enterprise-specific) join processes:

- Pre-provisioning the computer object so that the join user requires even fewer permissions and is not automatically added as the owner of the new machine account.


Administrator: Command Prompt
sys
File
Home
Share
View
This PC › Local Disk (C:) › sys
Quick access
Desktop
Downloads
Documents
Pictures
System32
This PC
3D Objects
Desktop
Documents
Downloads
Music
Name
Date modified
CL4711.join
24.08.2024 08:31
Active Directory Users and Computers
File
Action
View
Help
Active Directory Users and Com
Saved Queries
ita24.de
_Privileged
Builtin
Computers
Domain Controllers
ForeignSecurityPrincip
Keys
Lab
Clients
Name
Type
Description
CL4711
Computer
ITA24CL01
Computer

```
c:\>DJOIN /PROVISION /DOMAIN ita24.de /MACHINE CL4711 /MACHINEOU "OU=Clients,OU=Lab,DC=ita24,DC=de" /SAVEFILE c:\sys\CL4711.join

Provisioning the computer...
Successfully provisioned [CL4711] in the domain [ita24.de].
Provisioning data was saved successfully to [c:\sys\CL4711.join].

Computer provisioning completed successfully.
The operation completed successfully.
```

Figure 3: The DJOIN command can be used to limit somewhat the effect of domain joining.

- Pre-provisioning the computer object with an offline join (the `DJOIN` command [5]; Figure 3). In this case, you are not transferring credentials from a join account, but rather an offline join file, to the new computer. If you can find a secure and robust process, it is a very good approach.
- The use of a temporary domain admin account that you create for each new computer and delete again after deployment. This method should not be your preferred option, but it does work if all else fails.

Pre-provisioned computer objects have the added benefit that you can add them to the correct OU and group before installing the computer. Remember that Microsoft hardened the join process with an existing computer object in 2022 and again in 2023 [6], which limits your design options to some extent.

Another measure you should integrate into the deployment or domain join process is to remove the Domain Admins group from the Local Admins group of the newly minted AD members (Figure 4). In the default AD permissions configuration, you can easily use Group Policy Preferences or, less conveniently, the Restricted Groups group policy; however, the computer account must at least be able to see the Domain Admins group.

The same applies to the groups or users to which you want to grant Local Admins rights instead of the Domain Admins – the computer must see these objects to process the policy. One possible solution would be to add the pre-provisioned computer object temporarily to a group that can see the required privileged groups and then remove it from this group again after completing the deployment.

If your AD is already hybrid, and all users have a synchronized Entra ID account in addition to their AD account, you might want to consider not adding your client machines to AD at all, but joining them directly to Entra ID. Access to local resources can still be authenticated with Kerberos, provided you have a network connection to the DCs; however, access to services and systems that require NTLM would be ruled out. Of course, group policies are no longer available for the configuration, so you would have to use Intune or another tool, which is a small price to pay for the security gains.

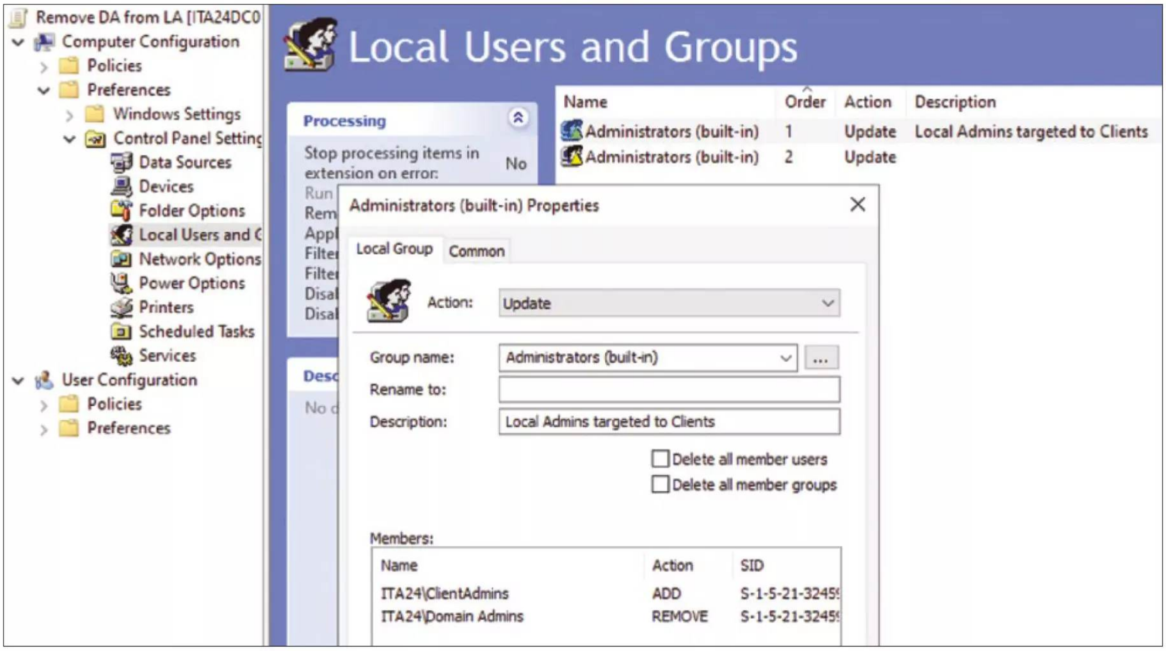
The Art of Transition

Secure AD design techniques are relatively easy to implement if you are building a new AD forest on a greenfield deployment. Even in a brownfield infrastructure, you can implement all the individual technical measures, but admins typically shy away from doing so because the side effects of the hardening and reorganization measures are unclear. Simply doing nothing in these troubled times is no longer an option, though, and rebuilding and migrating your AD can be the only escape route.

If you know what you are doing, nothing is wrong with migration, and a new forest is particularly advisable in the following situations:

- The current AD structure contains many domains in a forest, which you want to simplify to create a flat, single-domain topology; migration is inevitable in this case.
- The handling of DC system state backups has been too slack, meaning you have to assume that a potential attacker owns your DPAPI backup keys.

Even in forests where you have already made major changes to the authorization and delegation structures, it might


Remove DA from LA [ITA24DC0
Computer Configuration
Policies
Preferences
Windows Settings
Control Panel Setting
Data Sources
Devices
Folder Options
Local Users and C
Network Options
Power Options
Printers
Scheduled Tasks
Services
User Configuration
Policies
Preferences
Local Users and Groups
Processing
Stop processing items in extension on error:
No
Run
Rem
Appl
Filte
Filte
Disal
Disal
Desc
No d
Administrators (built-in) Properties
Local Group
Common

Name	Order	Action	Description
Administrators (built-in)	1	Update	Local Admins targeted to Clients
Administrators (built-in)	2	Update	

Action: Update

Group name: Administrators (built-in)

Rename to:

Description: Local Admins targeted to Clients

☐ Delete all member users

☐ Delete all member groups

Members:

Name	Action	SID
ITA24\ClientAdmins	ADD	S-1-5-21-3245
ITA24\Domain Admins	REMOVE	S-1-5-21-3245

Figure 4: Adding local administrator customizations to a policy.

be easier to migrate than embark on a painstaking clean-up with unpredictable consequences at each step. If you go down this route, you should try to migrate without adding SID History. If the planned coexistence of the old and new ADs in conjunction with existing applications makes SID History indispensable, you must at least have a clear roadmap for resolving the coexistence and removing SID History.

Conclusions

AD can be operated in a secure way, but you must have a general plan in place that is geared toward security and be willing to deviate massively from the default values delivered with AD and the machines involved. In this article, I could only scratch the surface and provide a rough direction toward account hygiene, including service accounts, cryptography and PKI, authorizations, and seemingly innocuous processes like joining new computers to AD. On a closing note, it's well worth investing time and effort in AD security, because AD will be with us for quite some time. ■

Info

[1] "Designing a Secure Active Directory" by Evgenij Smirnov, *ADMIN*, issue 86, 2025, pg. 86: [https://www.admin-magazine.com/Archive/2025/86/Designing-a-secure-Active-Directory]

[2] "Potatoes – Windows Privilege Escalation" by Jorge Lajara, November 2020: [https://jlajara.gitlab.io/Potatoes_Windows_Privesc]

[3] "Certified Pre-Owned" by Will Schroeder, *Medium*, June 2021: [https://posts.specterops.io/certified-pre-owned-d95910965cd2]

[4] "Configure deterministic 'good' for the online responder (OCSP)" by Uwe Gradenegger, September 2020: https://www.gradenegger.eu/en/configure-deterministic-good-for-the-onlineresponder-ocsp/

[5] Offline join: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/ff793312(v=ws.11)]

[6] "Netjoin: Domain join hardening changes" Microsoft KB5020276, last update August 2024: [https://support.microsoft.com/en-us/topic/kb5020276-netjoin-domain-join-hardening-changes-2b65a0f3-1f4c-42ef-ac0f-1caaf421baf8]

Author

Evgenij Smirnov has worked with computers since age 5 and delivered IT solutions for almost 30 years. His Active Directory and Exchange background led to PowerShell, which he's used and promoted since its first release. Evgenij is an active community lead at home in Berlin, a leading contributor to German online communities, a user group and conference speaker, a Microsoft Cloud and Datacenter Management MVP (2020), and author of *Building Modern Active Directory* (Apress, 2024).

Shop the Shop shop.linuxnewmedia.com/shop

Missed an issue?

You're in luck.

Most back issues are still available. Order now before they're gone!

shop.linuxnewmedia.com/shop

GET IT NOW!

SAVE TIME ON DELIVERY WITH OUR ALTERNATIVE PDF EDITIONS

ADMIN
Network & Security

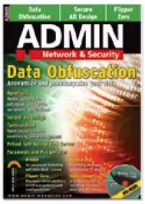
NEWSSTAND

Order online:
bit.ly/ADMIN-Library

ADMIN is your source for technical solutions to real-world problems. Every issue is packed with practical articles on the topics you need, such as: security, cloud computing, DevOps, HPC, storage, and more! Explore our full catalog of back issues for specific topics or to complete your collection.

#86 – March/April 2025

Data Obfuscation

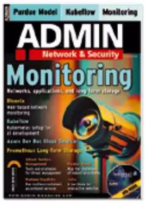


Generalization, suppression, perturbation, and differential privacy are essential data protection techniques that enable a balance between data security and usability and ensure compliance with legal requirements.

On the DVD: Rocky Linux 9.5 Minimal

#85 – January/February 2025

Monitoring

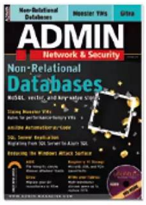


We look at infrastructure and systems monitoring and evaluate the tools needed to scale or store data in the long term for analysis.

On the DVD: Fedora Server 41

#84 – November/December 2024

Non-Relational Databases

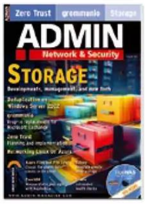


NoSQL databases manage data unsuitable for table-based relational databases and deliver efficiencies in scaling and high availability.

On the DVD: Ubuntu 24.04 LTS Server "Noble Numbat"

#83 – September/October 2024

Storage

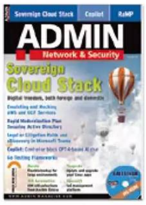


Improved management, suitable drivers, standardized protocol structures, and advancements in future-proof hardware and software lead to more durable, more manageable, and easier-to-repair storage.

On the DVD: TrueNAS SCALE 24.04 "Dragonfish"

#82 – July/August 2024

Sovereign Cloud Stack

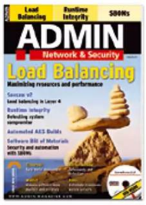


SCS liberates your data centers from monopolistic operations and companies beholden to out-country laws and regulations.

On the DVD: Kali Linux 2024.2

#81 – May/June 2024

Load Balancing



Load balancing on heavily frequented networks improves performance, availability, security, scalability, and the ability to handle peak loads.

On the DVD: SystemRescue 11.01

Lead Image © Author, 123RF.com

WRITE FOR US

Admin: Network and Security is looking for good, practical articles on system administration topics. We love to hear from IT professionals who have discovered innovative tools or techniques for solving real-world problems.

Tell us about your favorite:

- Interoperability solutions
- Practical tools for cloud environments
- Security problems and how you solved them
- Ingenious custom scripts
- Unheralded open source utilities
- Windows networking techniques that aren't explained (or aren't explained well) in the standard documentation

We need concrete, fully developed solutions: installation steps, configuration files, examples – we are looking for a complete discussion, not just a "hot tip" that leaves the details to the reader.

If you have an idea for an article, send a 1-2 paragraph proposal describing your topic to: *edit@admin-magazine.com*.

Authors

Amber Ankerholz	6
Chris Binnie	54
Andrea Ciarrocchi	26, 72
Chris Gibson	78
Ken Hess	3
Frank Hofmann	62
Rainer Keller	36
Philipp Koester	36
Martin Gerhard Loschwitz	18, 30, 46
Aryan Parashar	10
Thomas Reuß	68
Andreas Scheumann	36
Evgenij Smirnov	88
Matthias Wübbeling	58, 60

Contact Info

Editor in Chief
Joe Casad, jcasad@linuxnewmedia.com

Managing Editors
Rita L Sooby, rsooby@linuxnewmedia.com
Lori White, lwhite@linuxnewmedia.com

Senior Editor
Ken Hess

Localization & Translation
Ian Travis

News Editor
Amber Ankerholz

Copy Editors
Amy Pettle, Aubrey Vaughn

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen, Illustration based on graphics by xtrekx, 123RF.com

Advertising
Jessica Pryor, jpryor@linuxnewmedia.com

Publisher
Brian Osborn

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-876-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linuxnewmedia.com
www.admin-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the DVD provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2025 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media unless otherwise stated in writing.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by be1druckt GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN (Print ISSN: 2045-0702, Online ISSN: 2831-9583, USPS No: 347-931) is published bimonthly by Linux New Media USA, LLC, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. May/June 2025. Application to Mail at Periodicals Postage Prices is pending at Philadelphia, PA and additional mailing offices. POSTMASTER: send address changes to Linux Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Coming
Soon

ADMIN 88

Our next issue will be packed with all the great content you expect from *ADMIN*. Here are a few of the upcoming articles:

- Building a Port Scanner in Python
- Using Cyberduck for Object Storage
- Leveraging eBPF to Achieve Next-Generation Security **at** Runtime
- Route Tracers
- And much more!

Please note: Articles could change before the next issue.

Available Starting August Day 8

ADMIN Preview

ISSUE #83

Welcome to ADMIN Preview. This newsletter is a special reminder for all ADMIN readers that the latest issue of ADMIN Network & Security is available now.

Cover Story

Storage: Improved management, suitable drivers, standardized protocol structures, and advancements in future-proof hardware and software lead to more durable, more manageable, and easier-to-repair storage.

If you are an active digital subscriber, you should have received an email with instructions to download the latest issue.

Pay less for *ADMIN*! When you buy directly from us, you get the best price and receive your issues sooner.

BE THE FIRST TO SEE WHAT'S NEXT

Subscribe free to the *ADMIN* Preview newsletter and get a sneak peek at every article included in the next issue of *ADMIN*.

Sign up today at https://bit.ly/admin-preview

DrupalCon
Vienna 2025
14 - 17 October

Get Your Ticket to DrupalCon Vienna 2025!

Are you a developer, designer, project manager, agency leader, hosting provider, tech innovator, or student eager to dive into the world of Drupal?
Join us in Vienna for four exciting days of learning, connecting, and contributing!

What to Expect:

- 3 Conference Days packed with sessions in 7 tracks and BoFs
- Keynotes including Driesnote, Drupal CMS Spotlights, and The Web in 2025 - panel discussion
- Friday Contribution Day and Contribution Rooms throughout the week
- Recorded Sessions available after the event
- Social Events: Opening Reception, Trivia Night, Women in Drupal, and more
- International Splash Awards
- Networking with industry leaders, top companies, and fellow tech innovators

Single tickets

Early bird until 17 june	805€	Regular until 15 September	985€
Late until 16 October	1165€	Student until 16 October	485€

Group tickets

5 Early bird until 17 june	3900€	5 Regular until 15 September	4775€

Register today
and don't miss the biggest Drupal
event in Europe

GPU servers for demanding AI projects:

Maximum performance. Minimum price!

NVIDIA
RTX 6000
NVIDIA
NVIDIA
RTX 4000
NVIDIA

STARTING AT
$ 205
PER MONTH

HETZNER

Create your AI project with Hetzner GEX-Line servers!
Powered by NVIDIA GPUs with CUDA technology.

hetzner.com/gpu

20 GB VRAM
306.8 TFLOPS

48 GB VRAM
1457 TFLOPS

The perfect GPU server duo for every budget!

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change.
All rights reserved by the respective manufacturers.